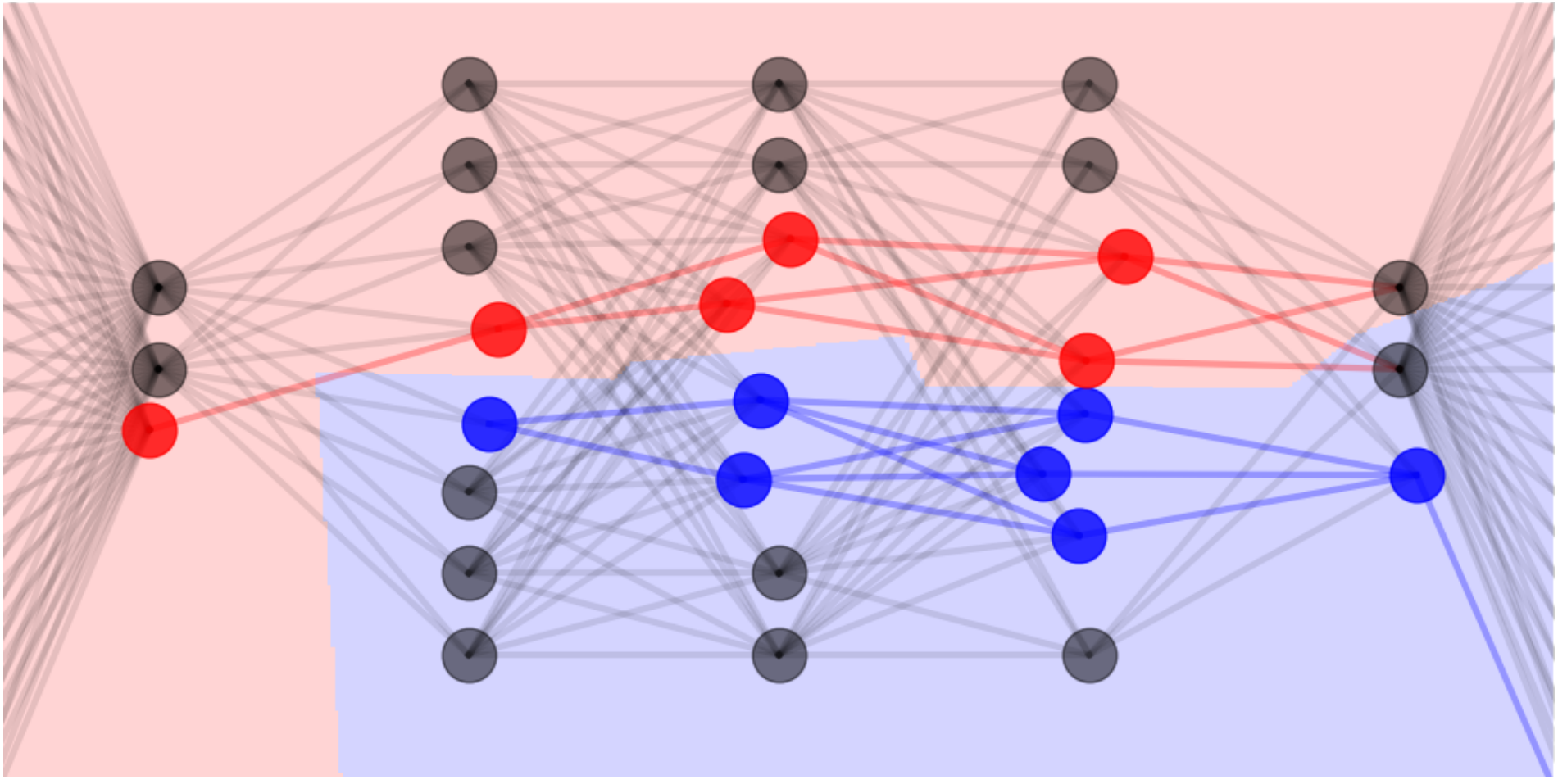
TU-Delft Machine and Deep Learning



RNNs

Lecture 6

Lecturer: Jan van Gemert

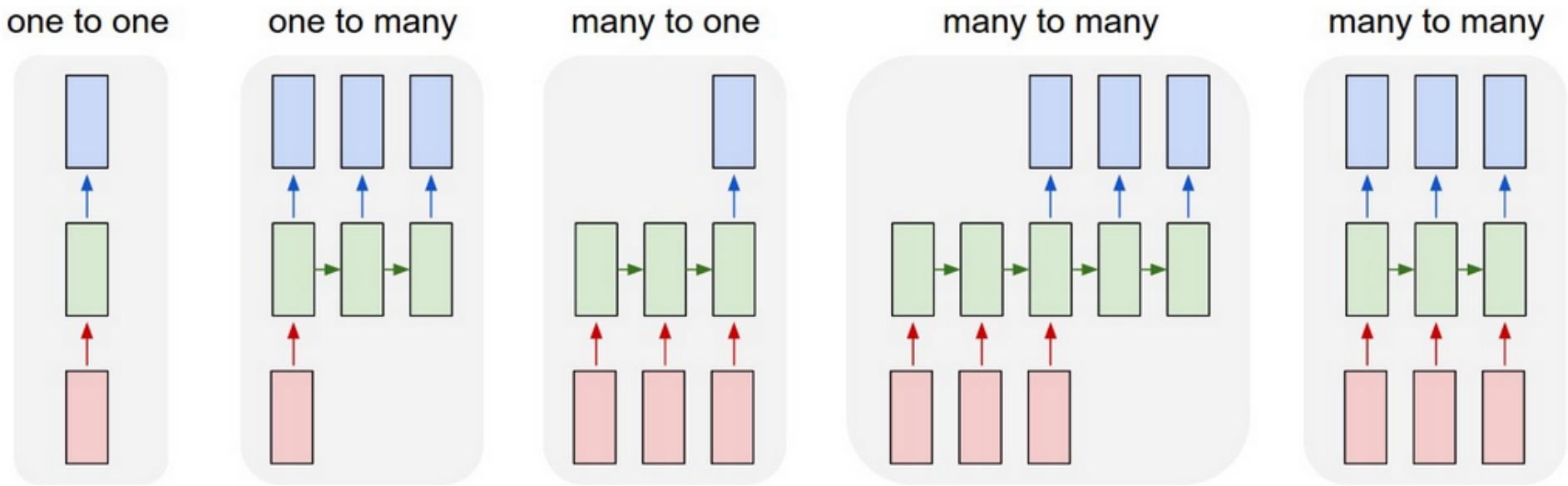
Topic: Recurrent networks

- RNNs
- Gradients in RNNs
- GRU and LSTM

Deep Learning book chapter 10;
D2L-book chapters: 9, 10

Types of sequential data

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



Sequences in the input, in the hidden layers, or in the output. [1]

[1] Andrej Karpathy's blog: "The Unreasonable Effectiveness of Recurrent Neural Networks"

Sequence data

Lets deep learn a language translation system.

Sequence data

Lets deep learn a language translation system.

Consider an example sentence, where each word is an input:

The student, after reading the book, was well prepared.

Sequence data

Lets deep learn a language translation system.

Consider an example sentence, where each word is an input:

The student, after reading the book, was well prepared.

Q: What are reasons to not use a feed forward MLP network?

Sequence data

Lets deep learn a language translation system.

Consider an example sentence, where each word is an input:

The student, after reading the book, was well prepared.

Q: What are reasons to not use a feed forward MLP network?

A: Sequences can have variable length.

Sequence data

Lets deep learn a language translation system.

Consider an example sentence, where each word is an input:

The student, after reading the book, was well prepared.

Q: What are reasons to not use a feed forward MLP network?

A: Sequences can have variable length.
Q: So what? Can't I just pad with zeros?

Sequence data

Lets deep learn a language translation system.

Consider an example sentence, where each word is an input:

The student, after reading the book, was well prepared.

Q: What are reasons to not use a feed forward MLP network?

A: Sequences can have variable length.

Q: So what? Can't I just pad with zeros?

A: No; Feed forward will learn exact word locations;

Sequence data

Lets deep learn a language translation system.

Consider an example sentence, where each word is an input:

The student, after reading the book, was well prepared.

Q: What are reasons to not use a feed forward MLP network?

A: Sequences can have variable length.
Q: So what? Can't I just pad with zeros?

A: No; Feed forward will learn exact word locations; Q: Why is that a problem?

Sequence data

Lets deep learn a language translation system.

Consider an example sentence, where each word is an input:

The student, after reading the book, was well prepared.

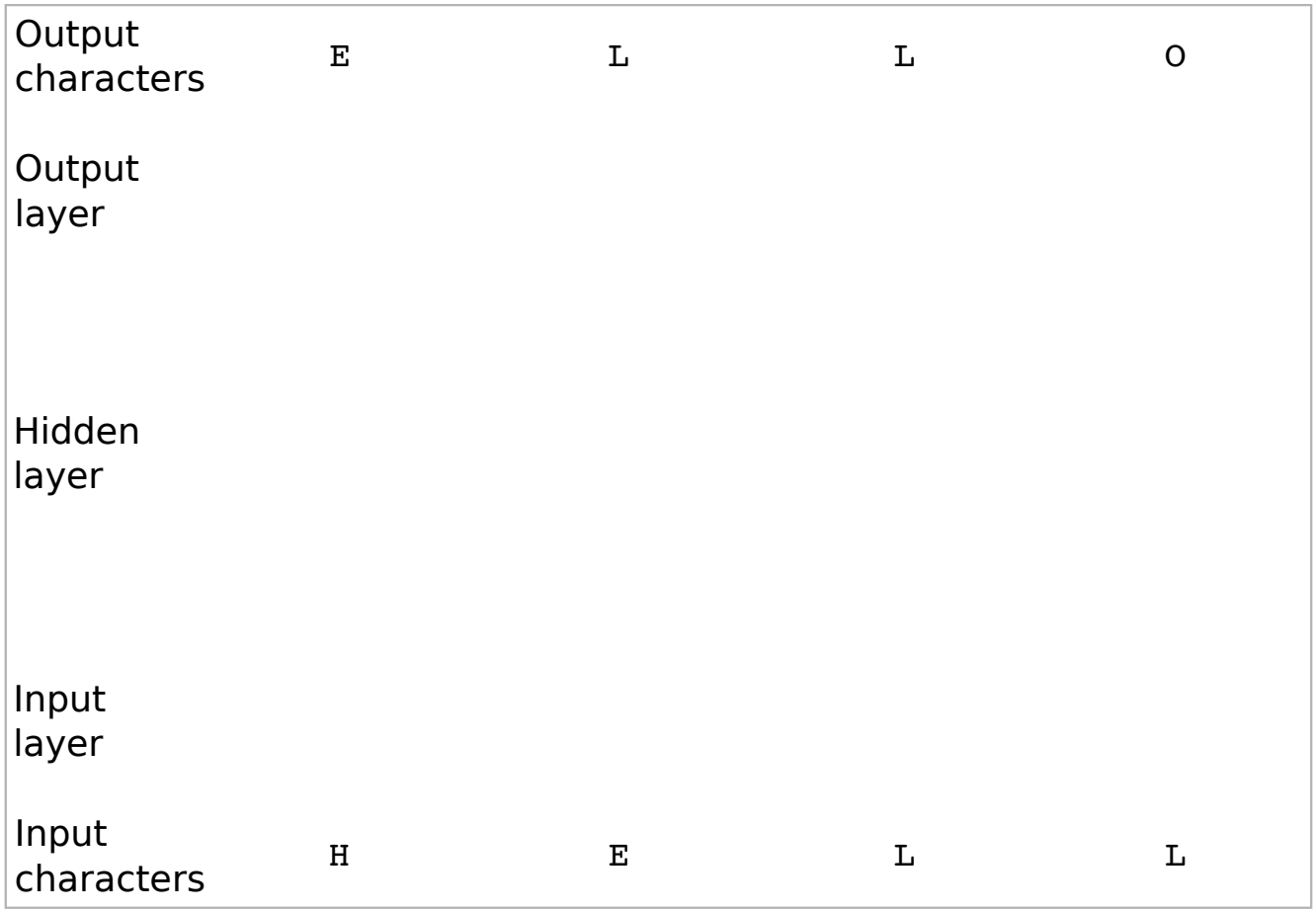
Q: What are reasons to not use a feed forward MLP network?

A: Sequences can have variable length.
Q: So what? Can't I just pad with zeros?

A: No; Feed forward will learn exact word locations; Q: Why is that a problem?
A: Need to see all possible word/sentence options during training (impossible).

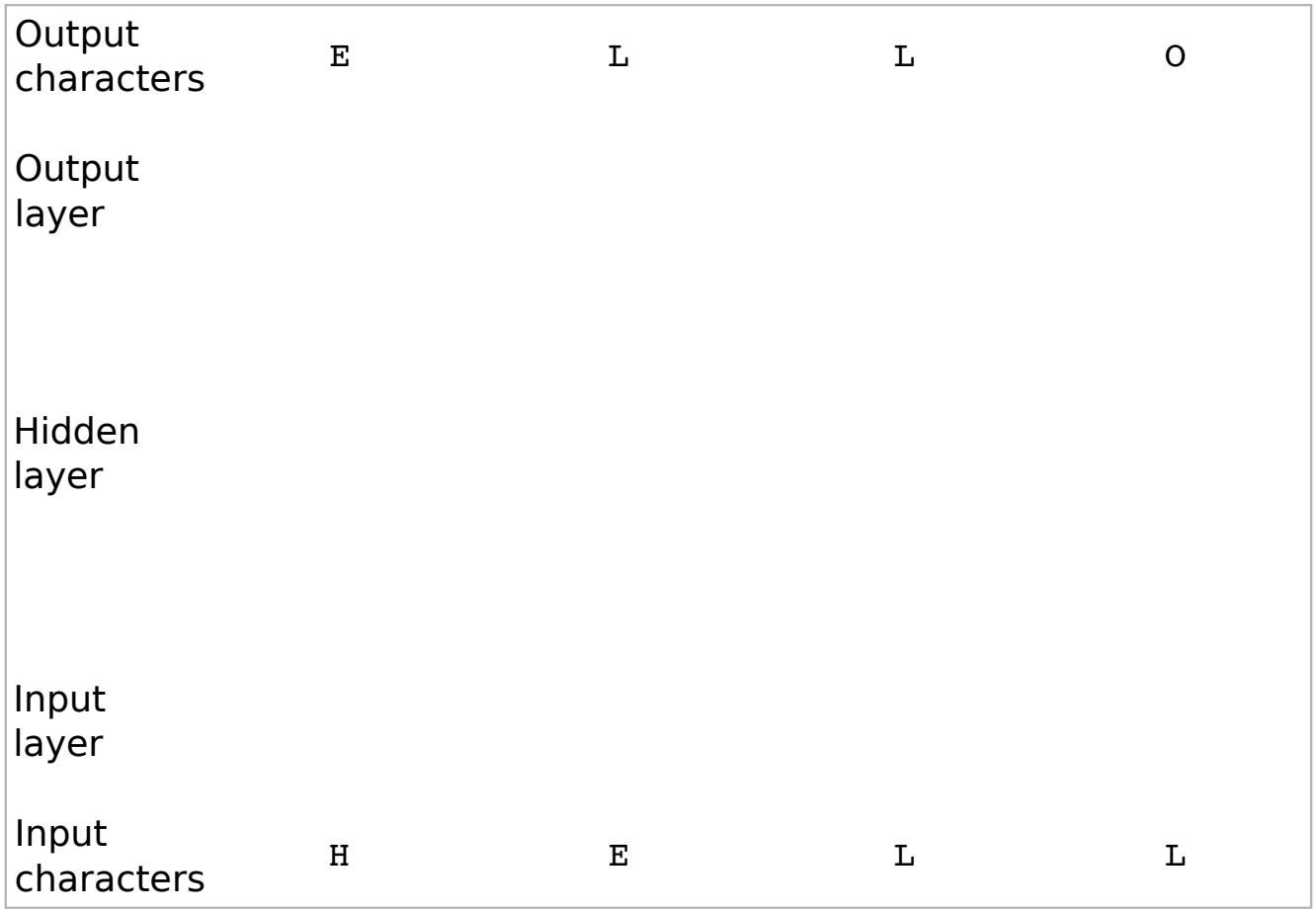
Example: Predict the next character

DL book, Chapter 10; and: `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`



Example: Predict the next character

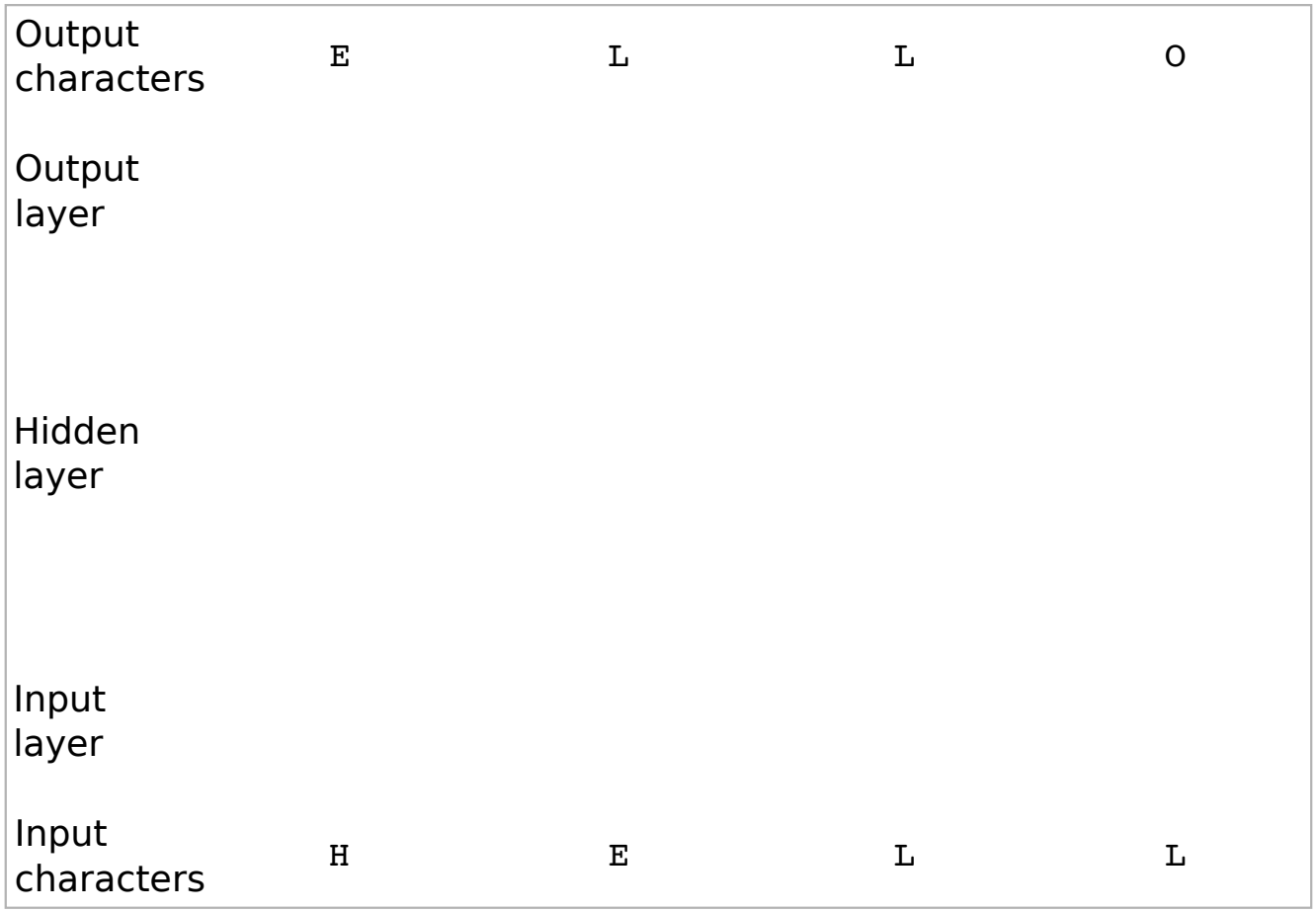
DL book, Chapter 10; and: `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`



Q: Explain the layout?

Example: Predict the next character

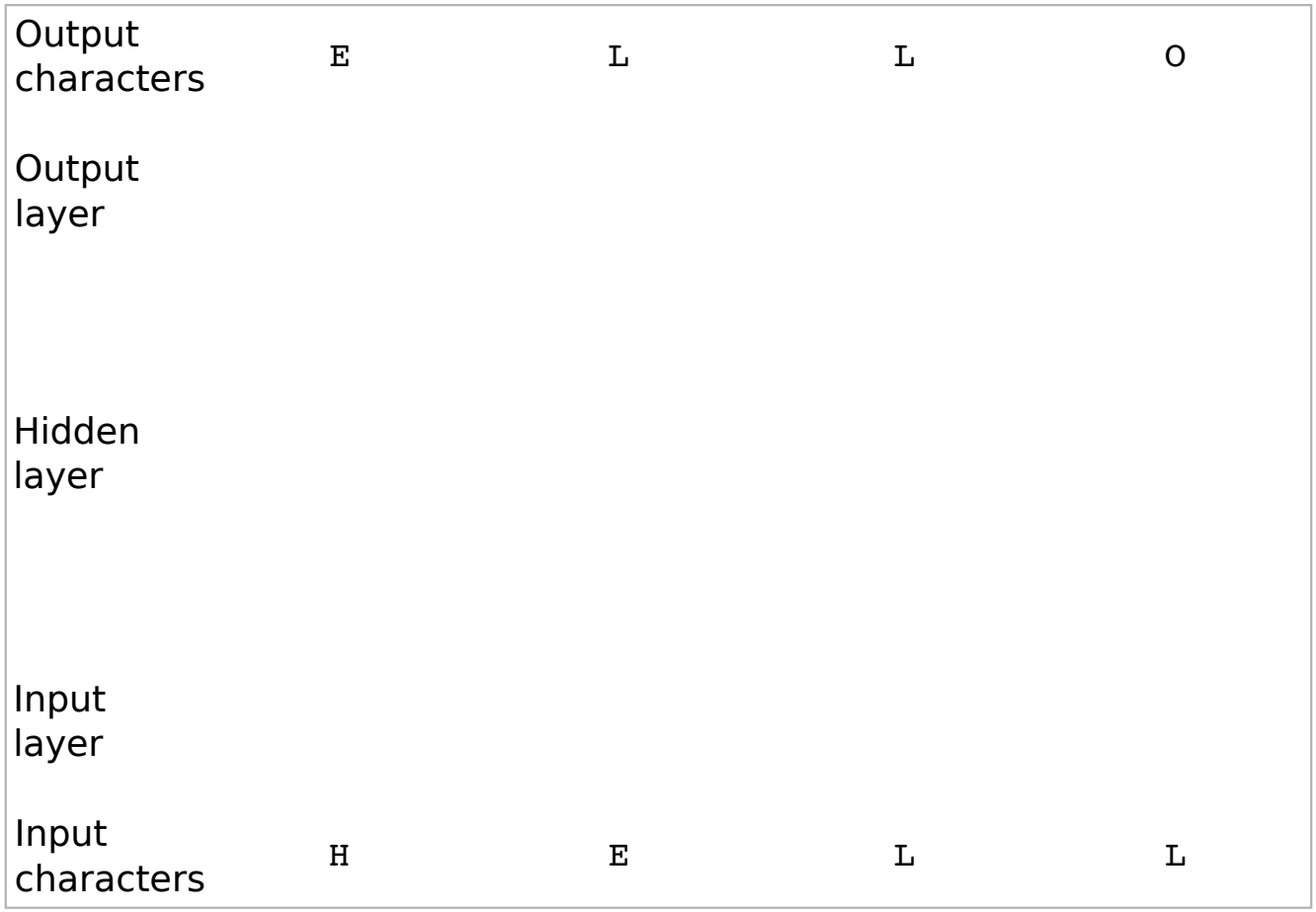
DL book, Chapter 10; and: `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`



Q: Explain the layout? A: A 2-layer network for letter prediction.

Example: Predict the next character

DL book, Chapter 10; and: `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`

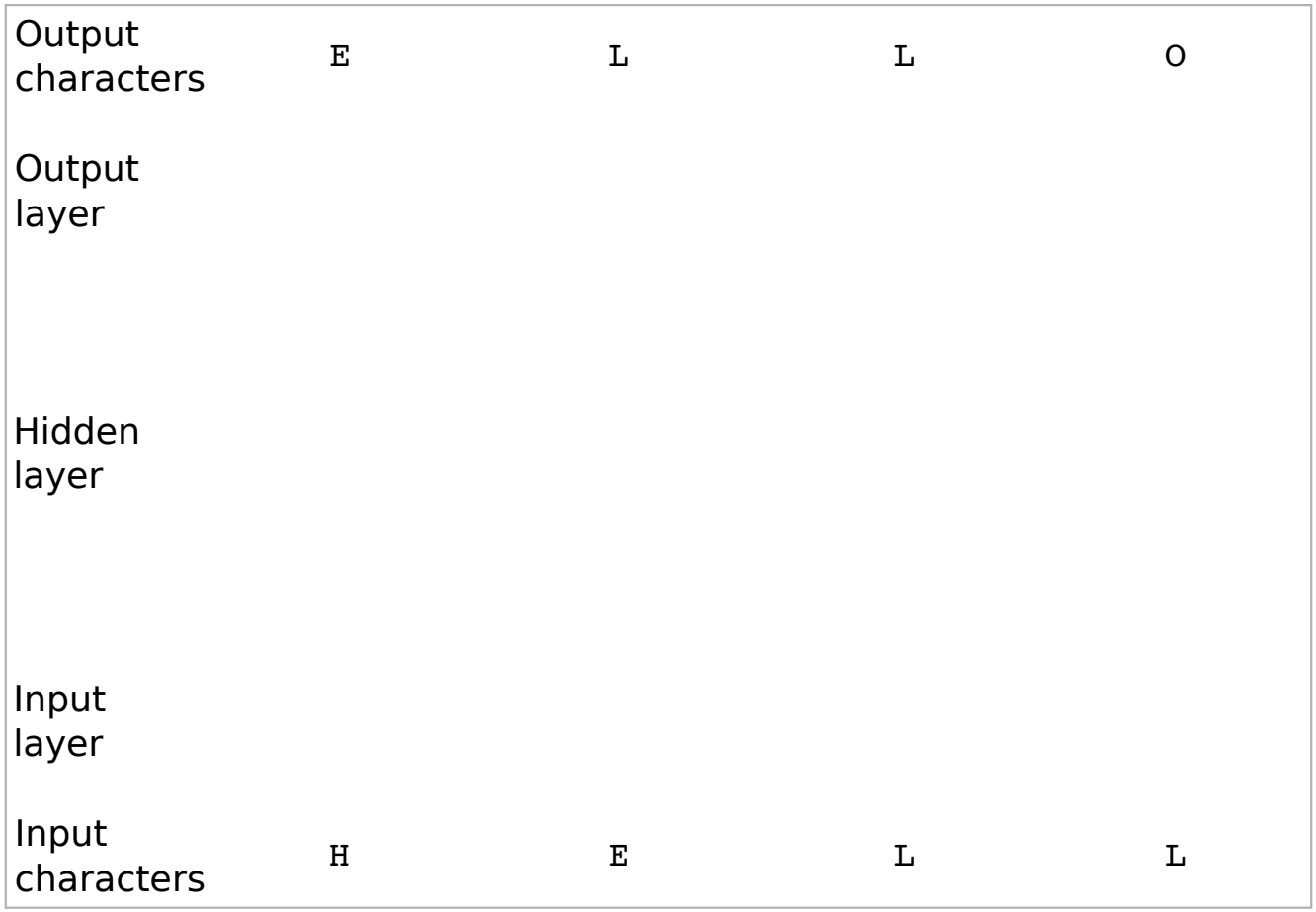


Q: Explain the layout? A: A 2-layer network for letter prediction.

Q: How to represent input letters?

Example: Predict the next character

DL book, Chapter 10; and: `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`

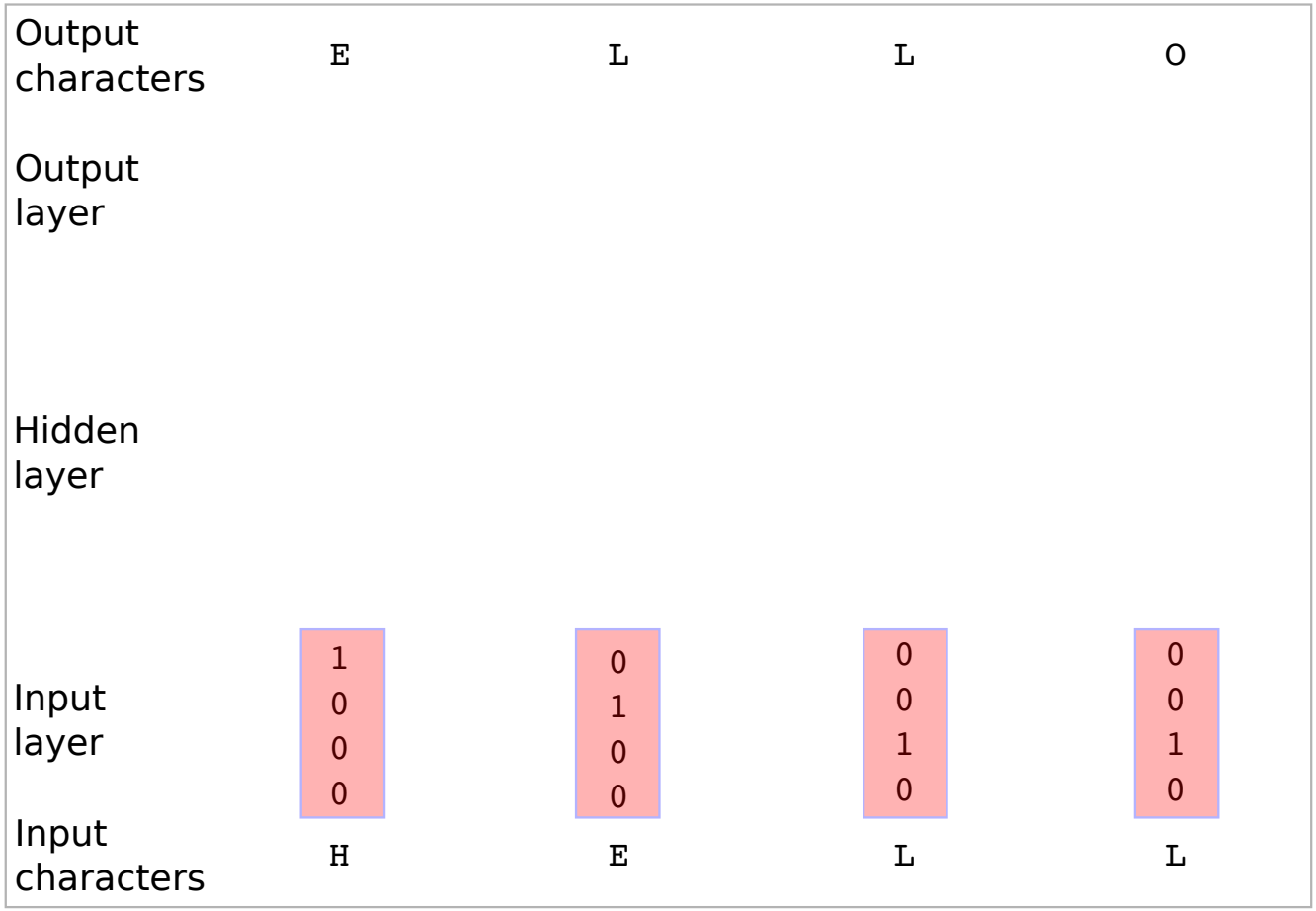


Q: Explain the layout? A: A 2-layer network for letter prediction.

Q: How to represent input letters? A: One-Hot encoding.

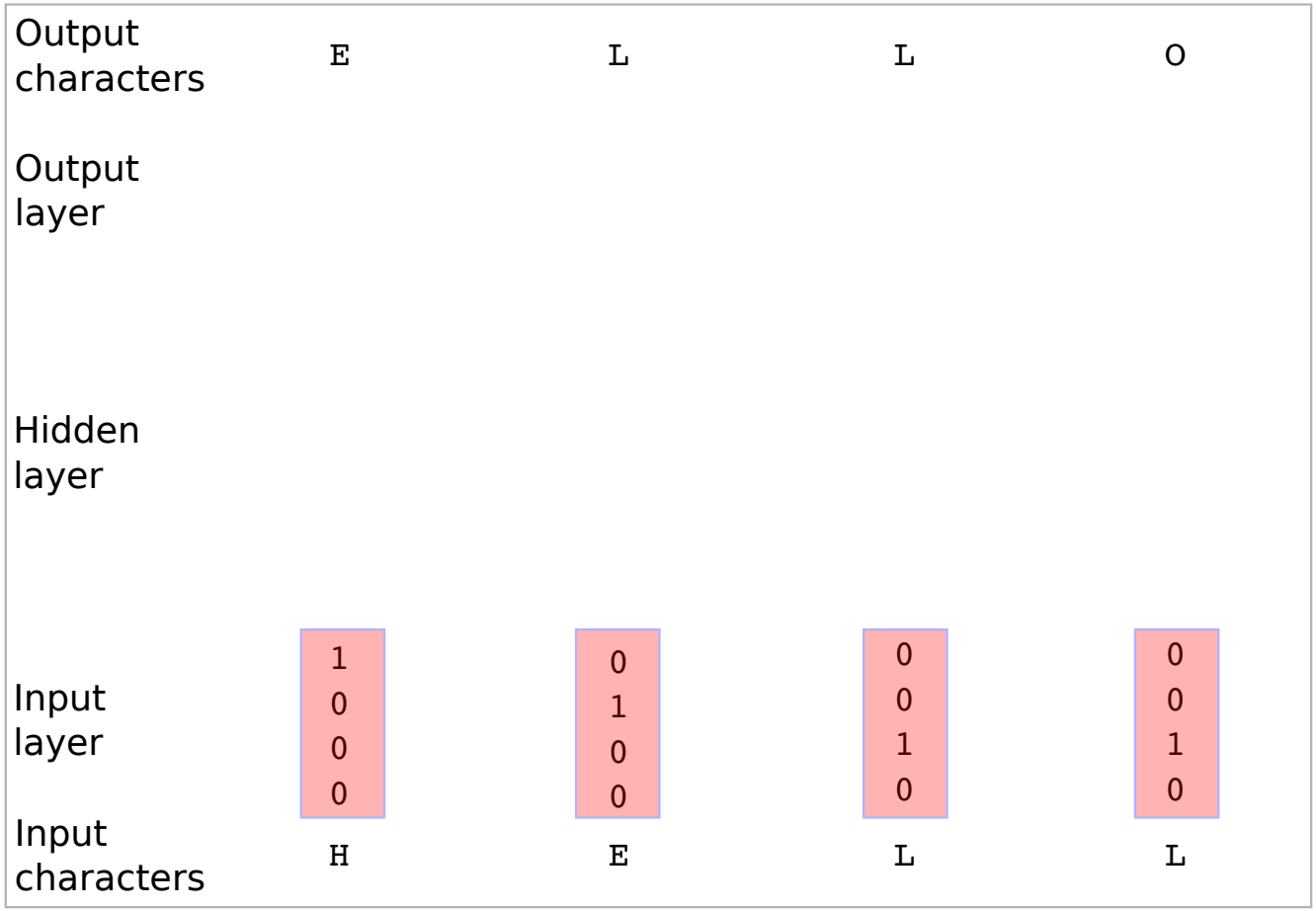
Example: Predict the next character

DL book, Chapter 10; and: `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`



Example: Predict the next character

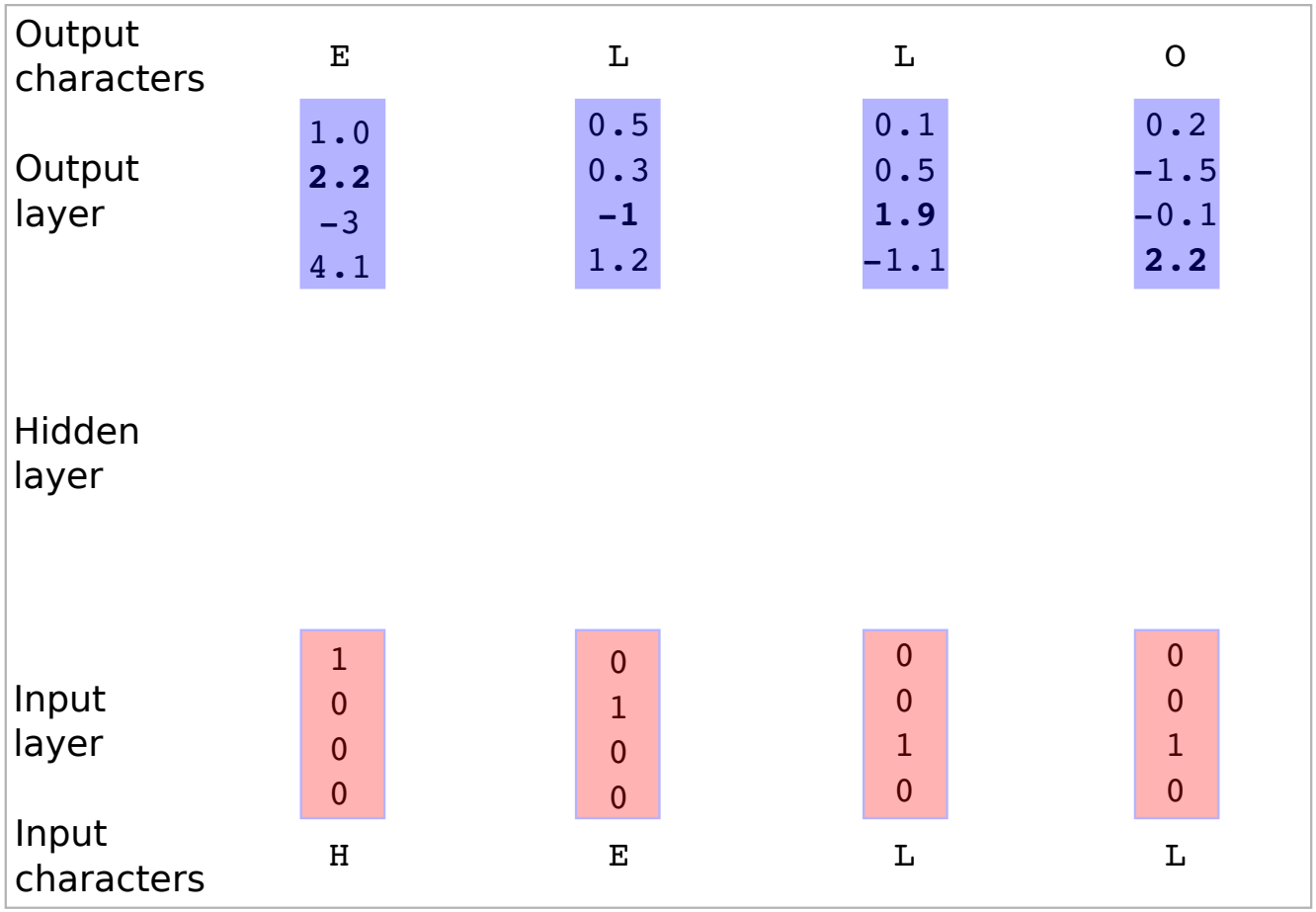
DL book, Chapter 10; and: `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`



Q: How to represent output?

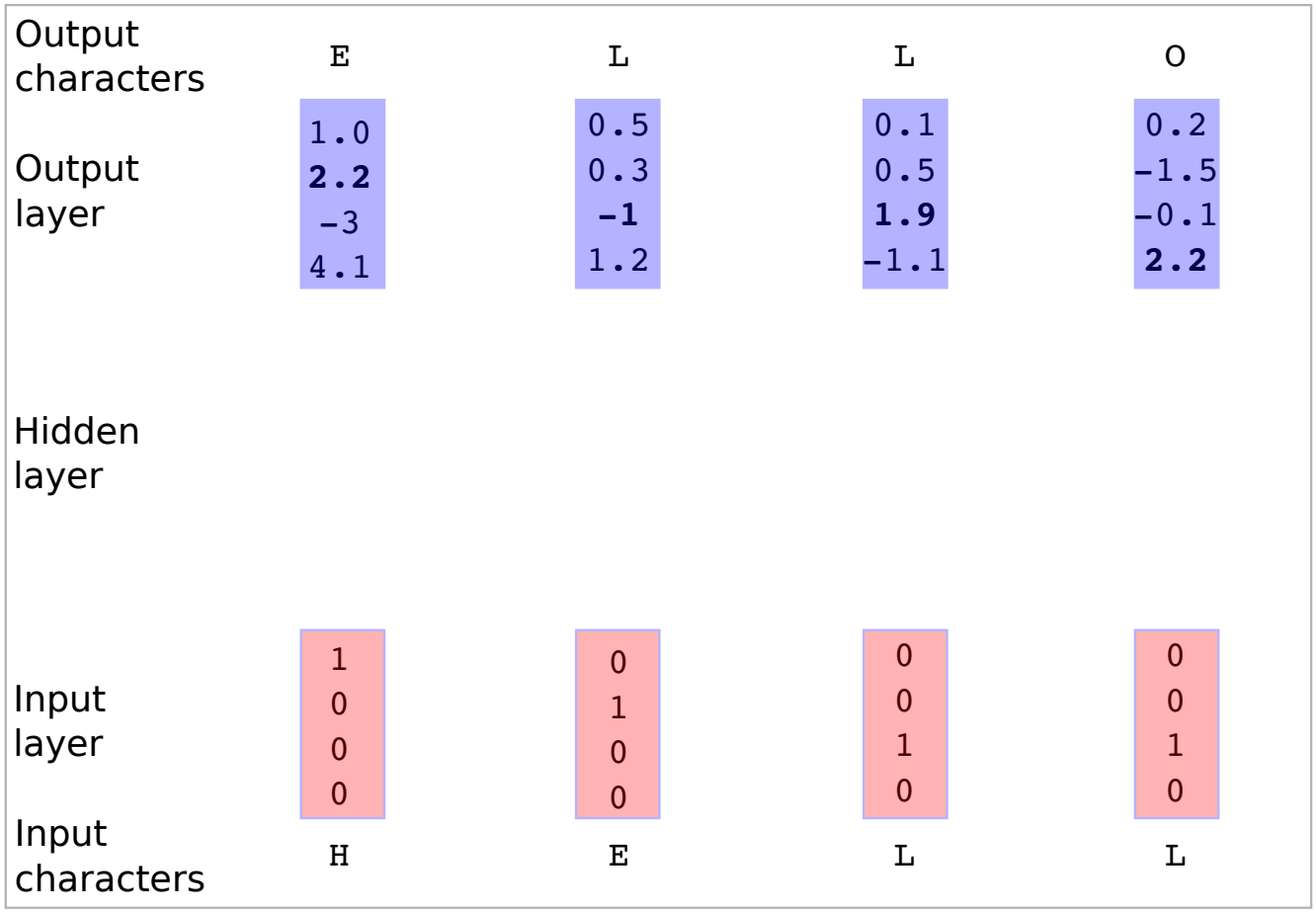
Example: Predict the next character

DL book, Chapter 10; and: `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`



Example: Predict the next character

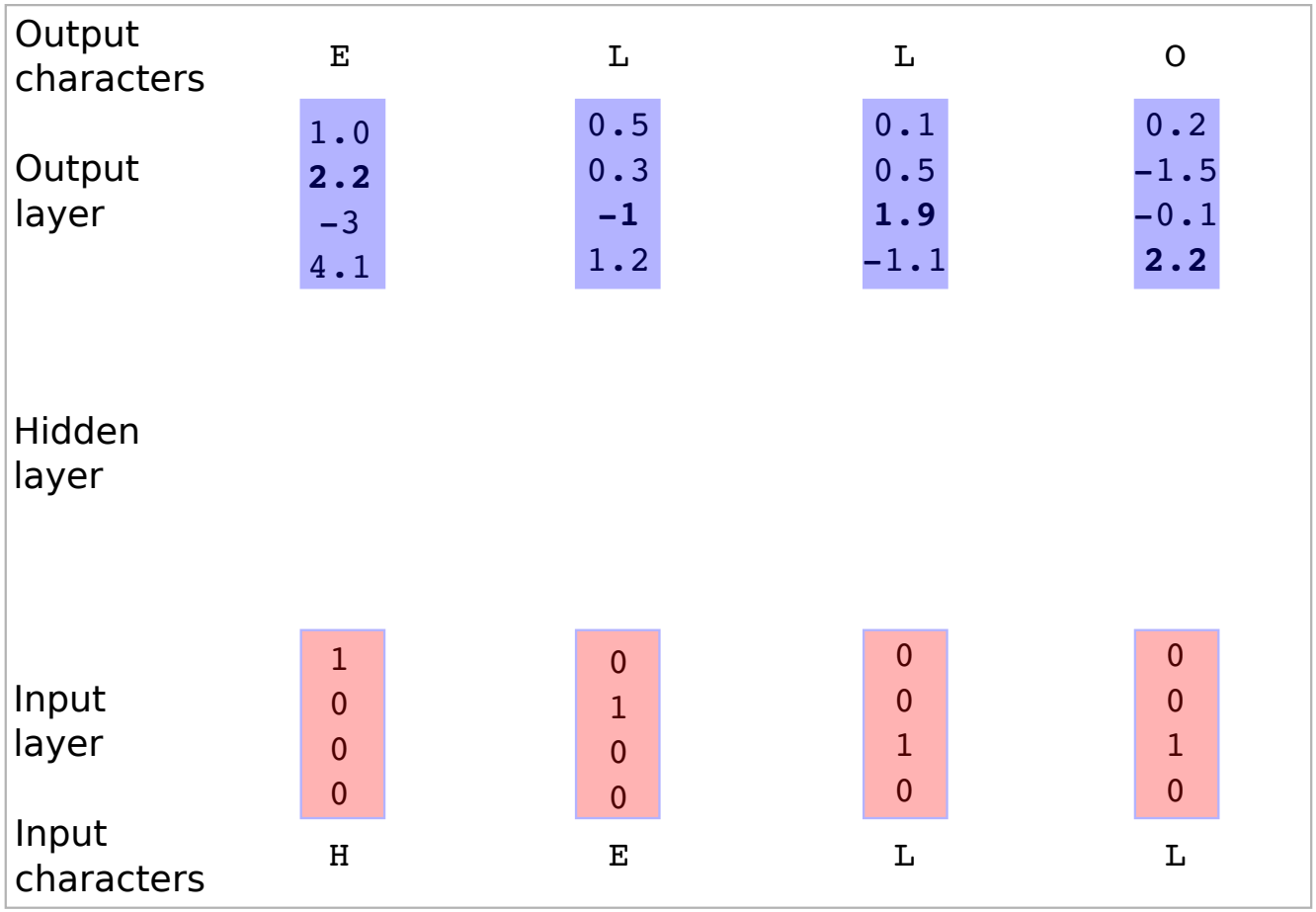
DL book, Chapter 10; and: `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`



Q: What do the bold numbers mean?

Example: Predict the next character

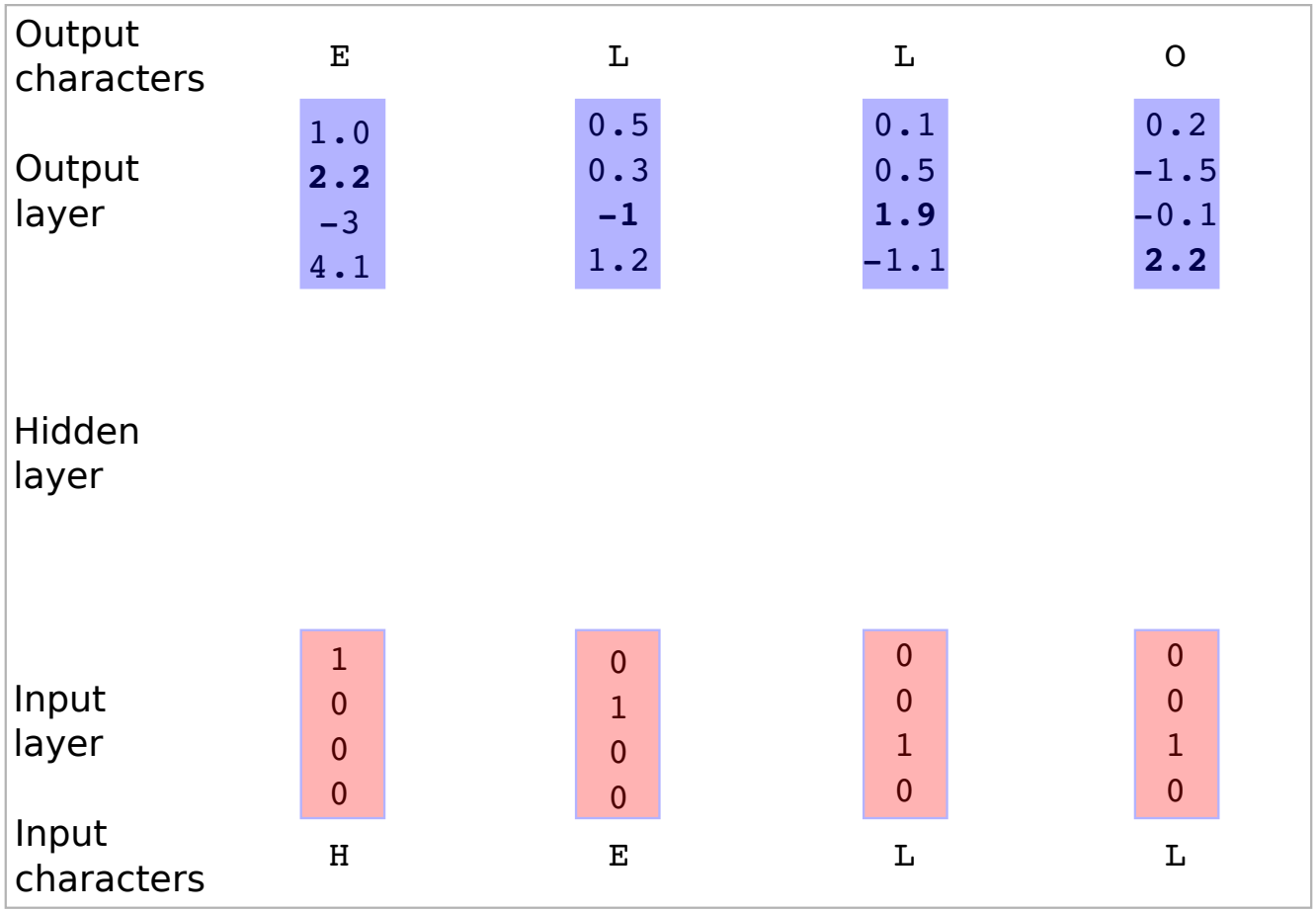
DL book, Chapter 10; and: `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`



Q: What do the bold numbers mean? A: Target (make high)

Example: Predict the next character

DL book, Chapter 10; and: `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`

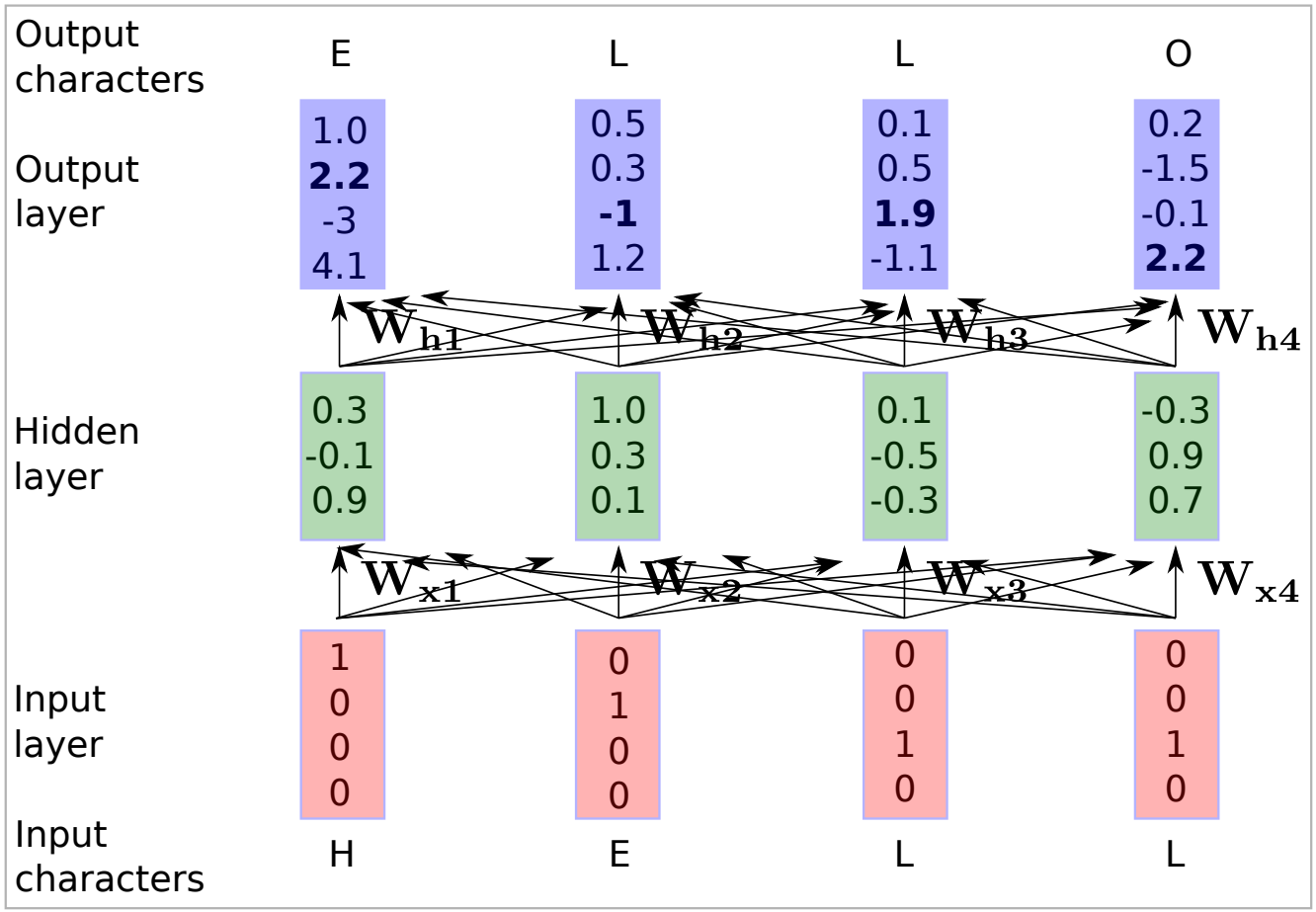


Q: What do the bold numbers mean? A: Target (make high)

Q: What would a fully connected network look like?

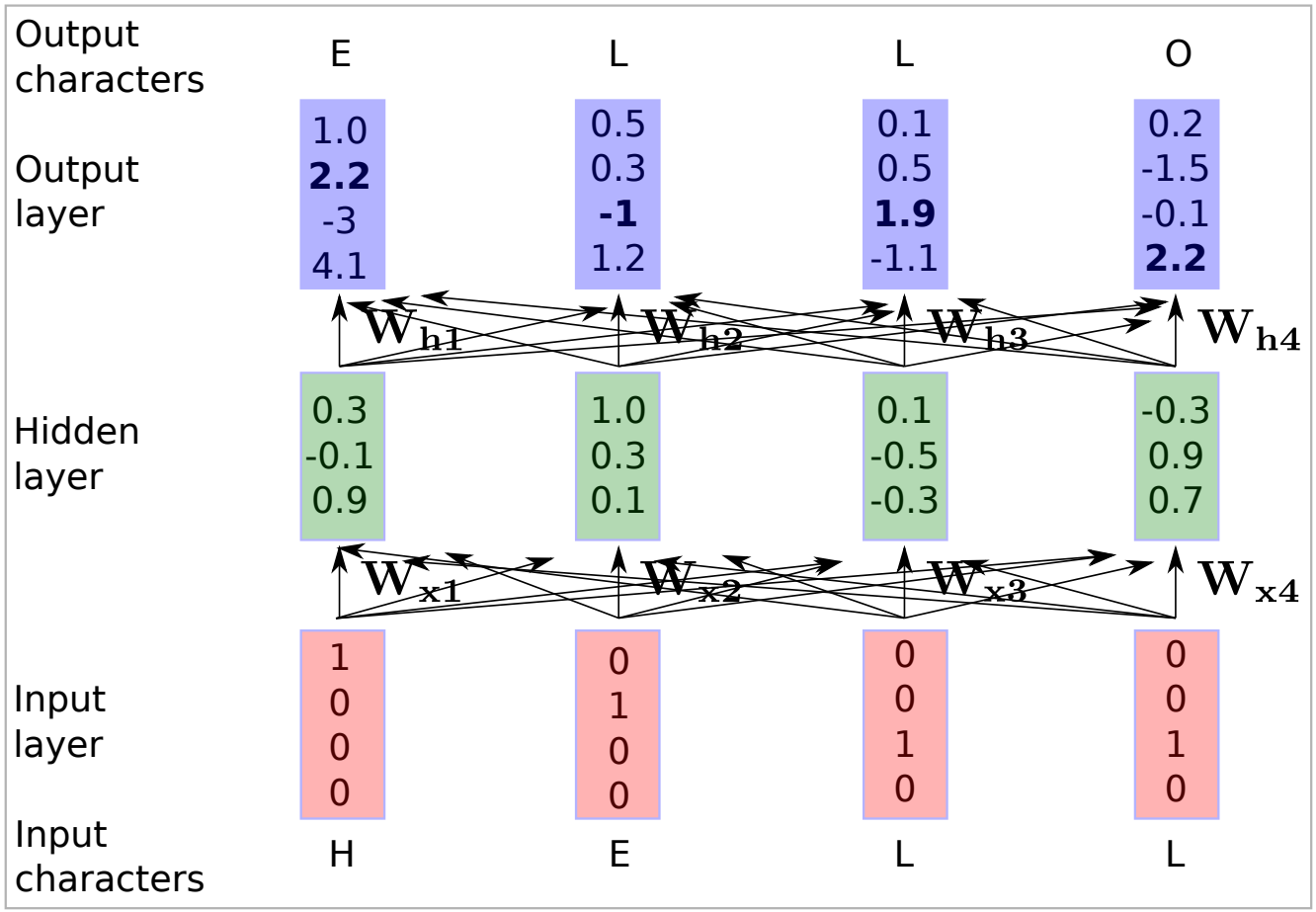
Example: Predict the next character

DL book, Chapter 10; and: `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`



Example: Predict the next character

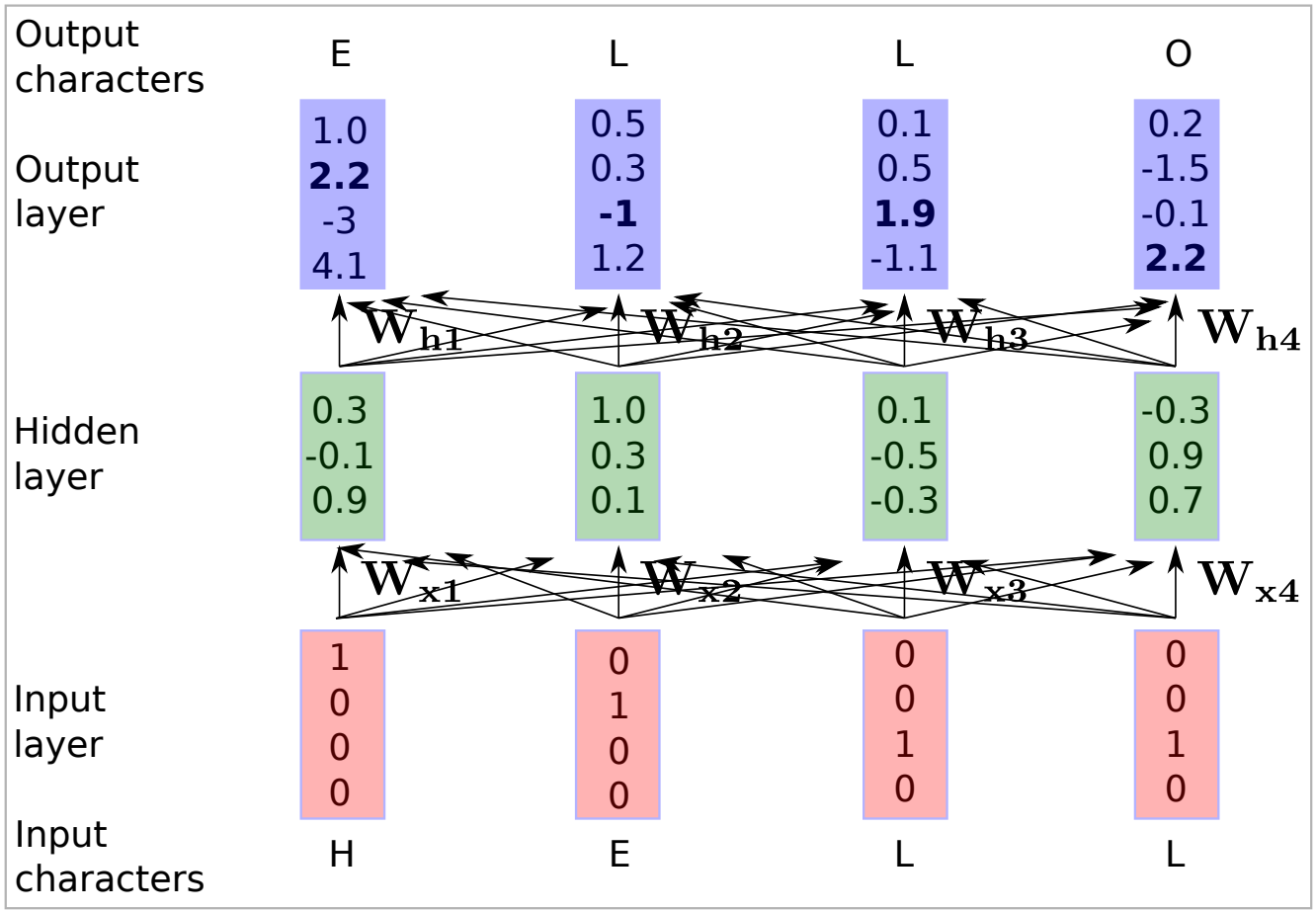
DL book, Chapter 10; and: `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`



Q: Easy solution to remove position-specific weights?

Example: Predict the next character

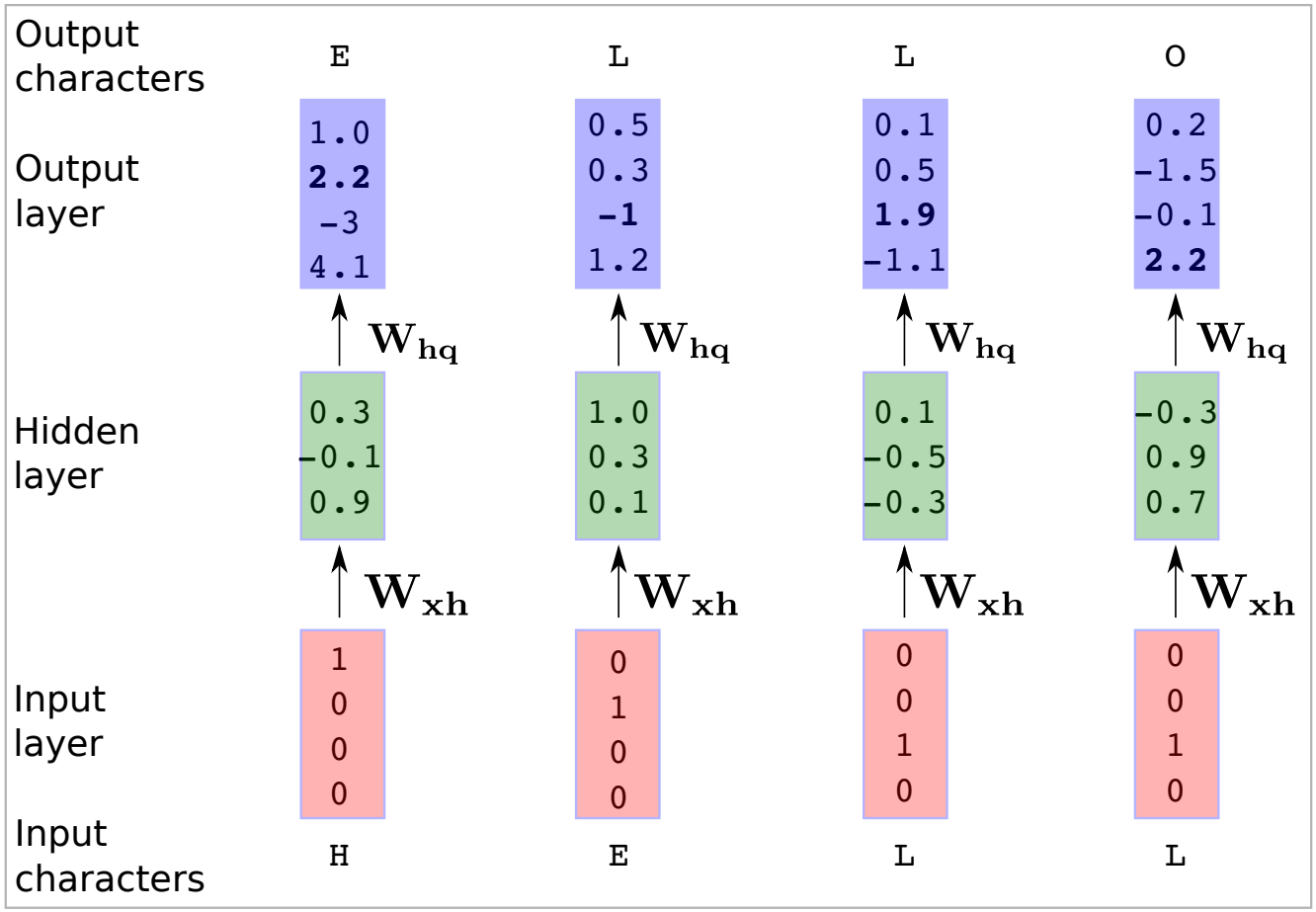
DL book, Chapter 10; and: `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`



Q: Easy solution to remove position-specific weights? A: Share all weights (convolution).

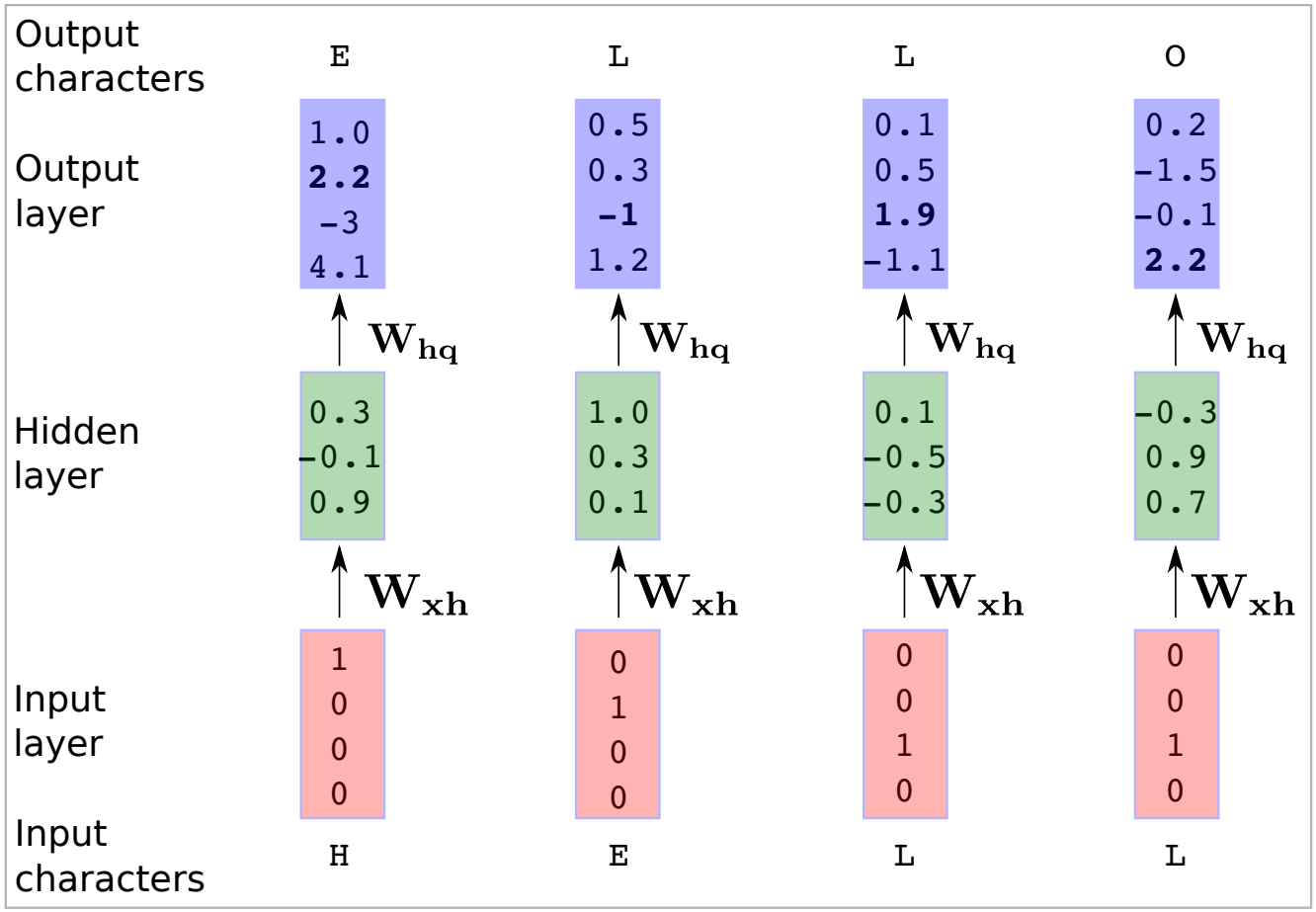
Example: Predict the next character

DL book, Chapter 10; and: `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`



Example: Predict the next character

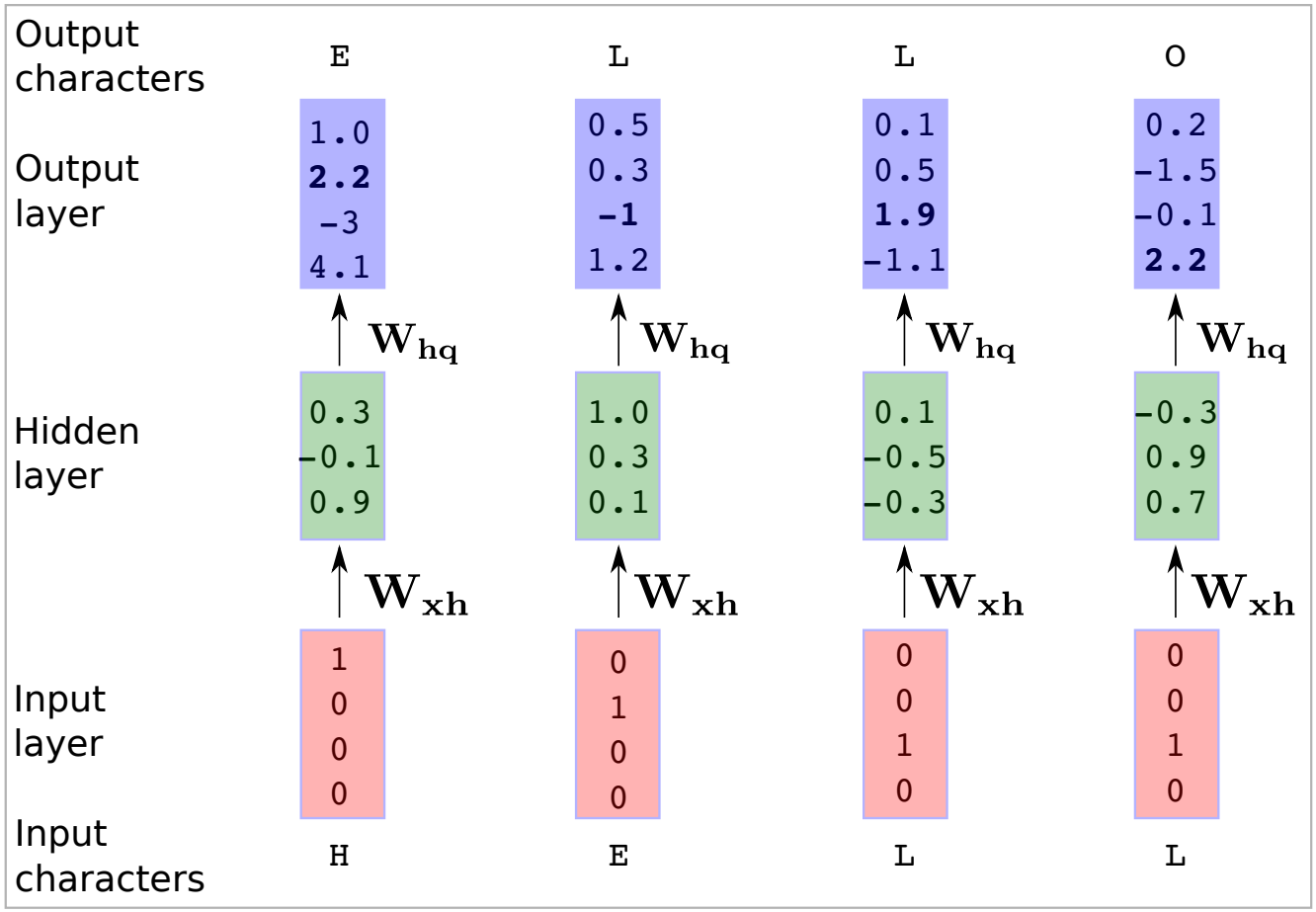
DL book, Chapter 10; and: `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`



Q: What is the problem for 'L'?

Example: Predict the next character

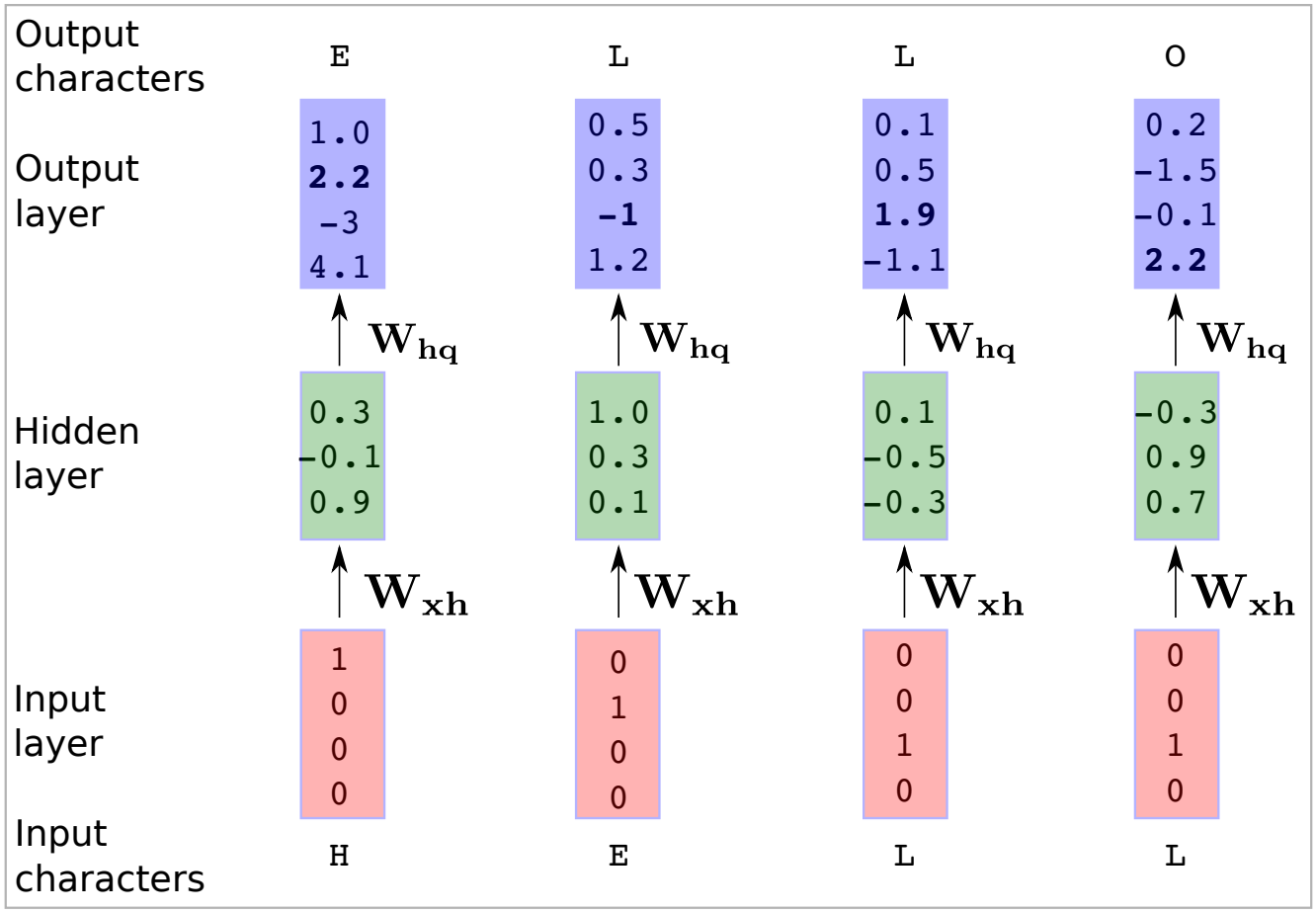
DL book, Chapter 10; and: `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`



Q: What is the problem for 'L'? A: Individual predictions. No past/context.

Example: Predict the next character

DL book, Chapter 10; and: `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`

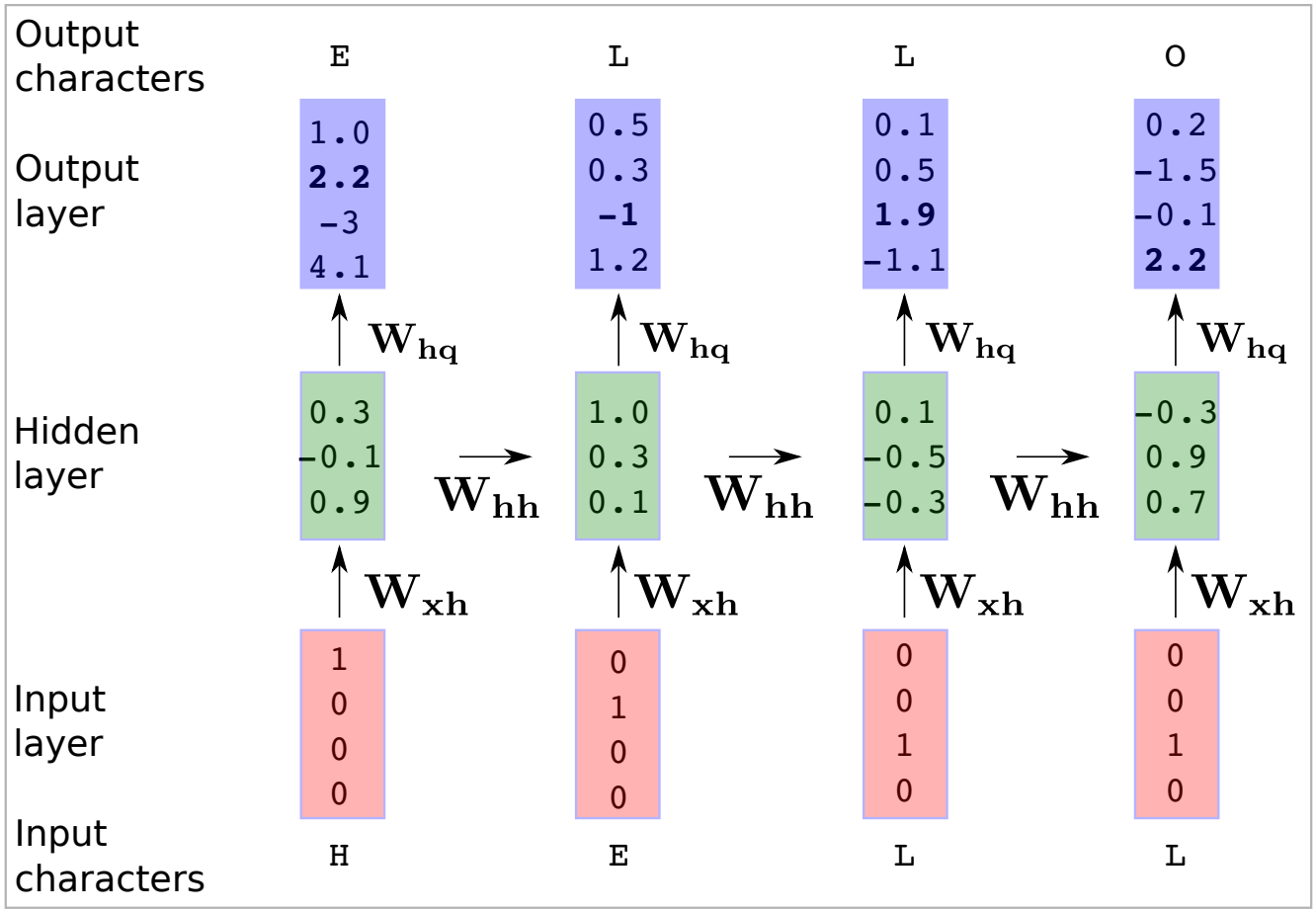


Q: What is the problem for 'L'? A: Individual predictions. No past/context.

Q: How can the past be included?

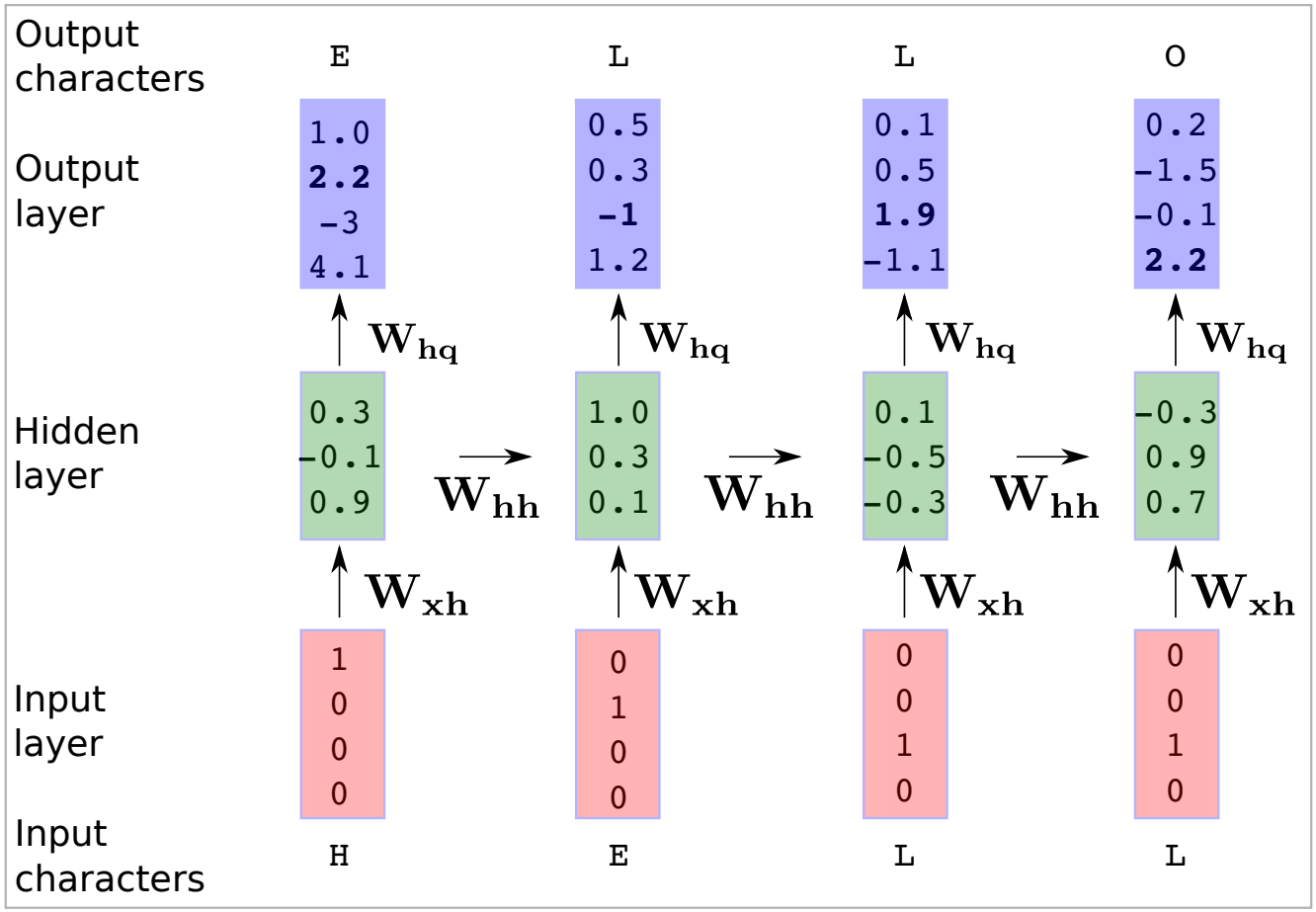
Example: Predict the next character

DL book, Chapter 10; and: `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`



Example: Predict the next character

DL book, Chapter 10; and: `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`



Recurrent Neural Network:

Process a sequence in order and share the same parameters at each time step.

RNN

https://d2l.ai/chapter_recurrent-neural-networks/rnn.html#recurrent-networks-with-hidden-states

Recurrent Neural Network:
Process a sequence in order and share the same parameters at each time step.

RNN

https://d2l.ai/chapter_recurrent-neural-networks/rnn.html#recurrent-networks-with-hidden-states

Recurrent Neural Network:
Process a sequence in order and share the same parameters at each time step.

For a sequence $\mathbf{X_t}$ for $t = 1, 2, 3, \ldots, T$

RNN

https://d2l.ai/chapter_recurrent-neural-networks/rnn.html#recurrent-networks-with-hidden-states

Recurrent Neural Network:
Process a sequence in order and share the same parameters at each time step.

For a sequence $\mathbf{X_t}$ for $t = 1, 2, 3, \ldots, T$

$$\mathbf{H_t} = \phi(\mathbf{X_t}\mathbf{W_{xh}} + \mathbf{H_{t-1}}\mathbf{W_{hh}} + \mathbf{b_h}),$$

RNN

https://d2l.ai/chapter_recurrent-neural-networks/rnn.html#recurrent-networks-with-hidden-states

Recurrent Neural Network:
Process a sequence in order and share the same parameters at each time step.

For a sequence $\mathbf{X_t}$ for $t = 1, 2, 3, \ldots, T$

$$\mathbf{H_t} = \phi(\mathbf{X_t}\mathbf{W_{xh}} + \mathbf{H_{t-1}}\mathbf{W_{hh}} + \mathbf{b_h}), \qquad \text{output } \mathbf{O_t} = \mathbf{H_t}\mathbf{W_{hq}} + \mathbf{b_q}$$

RNN

https://d2l.ai/chapter_recurrent-neural-networks/rnn.html#recurrent-networks-with-hidden-states

Recurrent Neural Network:
Process a sequence in order and share the same parameters at each time step.

For a sequence $\mathbf{X_t}$ for $t = 1, 2, 3, \ldots, T$

$$\mathbf{H_t} = \phi(\mathbf{X_t}\mathbf{W_{xh}} + \mathbf{H_{t-1}}\mathbf{W_{hh}} + \mathbf{b_h}), \qquad \text{output } \mathbf{O_t} = \mathbf{H_t}\mathbf{W_{hq}} + \mathbf{b_q}$$

Q: Explain the symbols?

RNN

https://d2l.ai/chapter_recurrent-neural-networks/rnn.html#recurrent-networks-with-hidden-states

Recurrent Neural Network:
Process a sequence in order and share the same parameters at each time step.

For a sequence $\mathbf{X_t}$ for $t = 1, 2, 3, \ldots, T$

$$\mathbf{H_t} = \phi(\mathbf{X_t}\mathbf{W_{xh}} + \mathbf{H_{t-1}}\mathbf{W_{hh}} + \mathbf{b_h}), \qquad \text{output } \mathbf{O_t} = \mathbf{H_t}\mathbf{W_{hq}} + \mathbf{b_q}$$

Q: Explain the symbols?

- $\mathbf{H_t}$: Activation at current time step t
- ϕ: non-linearity, (often a $\tanh$, range in (-1,1))
- $\mathbf{X_t}$: input at time t
- $\mathbf{W_{xh}}$: Learned weights for input at time t
- $\mathbf{H_{t-1}}$: Activation at previous time step $t-1$
- $\mathbf{W_{hh}}$: Learned weights: how to use the previous information at $t-1$
- $\mathbf{W_{hq}}$: Learned weights for output at time t
- $\mathbf{b_h}, \mathbf{b_q}$: Learned bias terms

RNN: Nr parameters doesn't grow with sequence length

https://d2l.ai/chapter_recurrent-neural-networks/rnn.html#recurrent-networks-with-hidden-states

Recurrent Neural Network:
Process a sequence in order and share the same parameters at each time step.

For a sequence $\mathbf{X_t}$ for $t = 1, 2, 3, \ldots, T$

$$\mathbf{H_t} = \phi(\mathbf{X_t}\mathbf{W_{xh}} + \mathbf{H_{t-1}}\mathbf{W_{hh}} + \mathbf{b_h}), \qquad \text{output } \mathbf{O_t} = \mathbf{H_t}\mathbf{W_{hq}} + \mathbf{b_q}$$

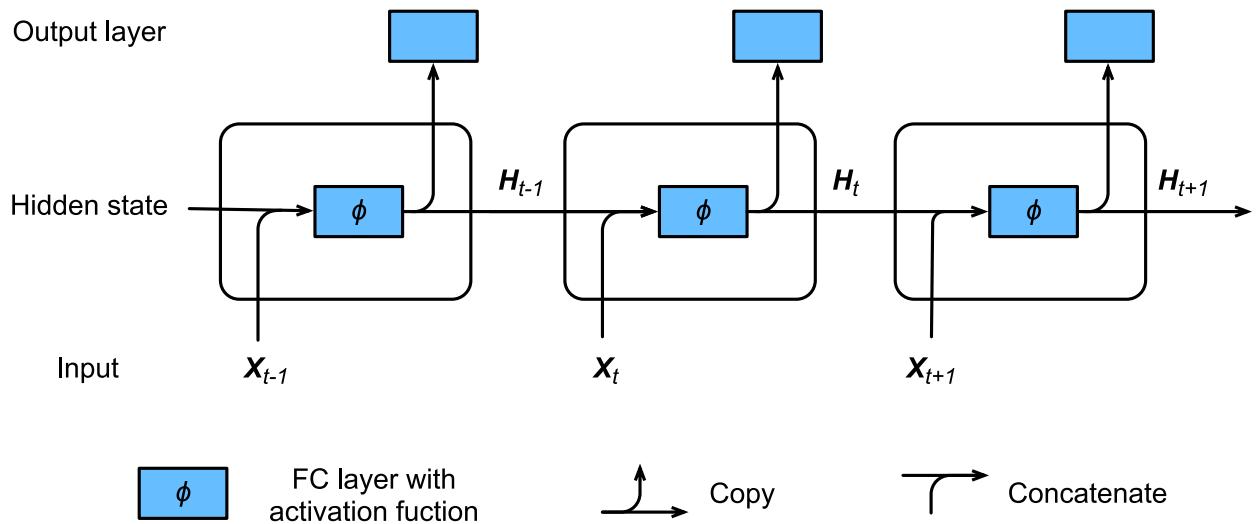
RNN: Nr parameters doesn't grow with sequence length

https://d2l.ai/chapter_recurrent-neural-networks/rnn.html#recurrent-networks-with-hidden-states

Recurrent Neural Network:
Process a sequence in order and share the same parameters at each time step.

For a sequence $\mathbf{X_t}$ for $t = 1, 2, 3, \ldots, T$

$$\mathbf{H_t} = \phi(\mathbf{X_t W_{xh}} + \mathbf{H_{t-1} W_{hh}} + \mathbf{b_h}), \qquad \text{output } \mathbf{O_t} = \mathbf{H_t W_{hq}} + \mathbf{b_q}$$



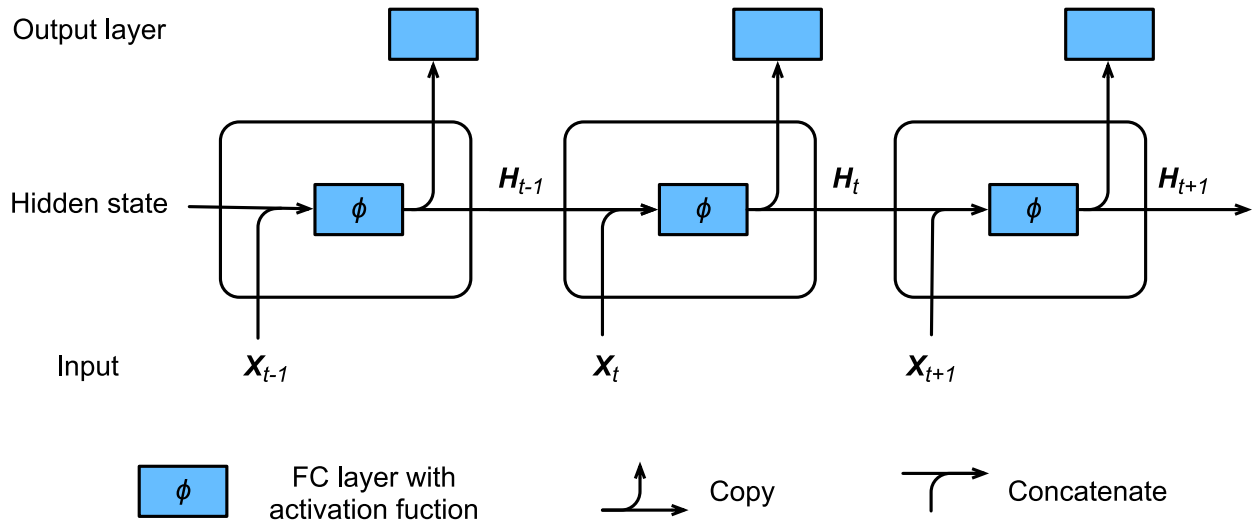
RNN: Nr parameters doesn't grow with sequence length

https://d2l.ai/chapter_recurrent-neural-networks/rnn.html#recurrent-networks-with-hidden-states

Recurrent Neural Network:
Process a sequence in order and share the same parameters at each time step.

For a sequence $\mathbf{X_t}$ for $t = 1, 2, 3, \ldots, T$

$$\mathbf{H_t} = \phi(\mathbf{X_t}\mathbf{W_{xh}} + \mathbf{H_{t-1}}\mathbf{W_{hh}} + \mathbf{b_h}), \qquad \text{output } \mathbf{O_t} = \mathbf{H_t}\mathbf{W_{hq}} + \mathbf{b_q}$$



Q: How does the "concatenate" operator relate to the equation?

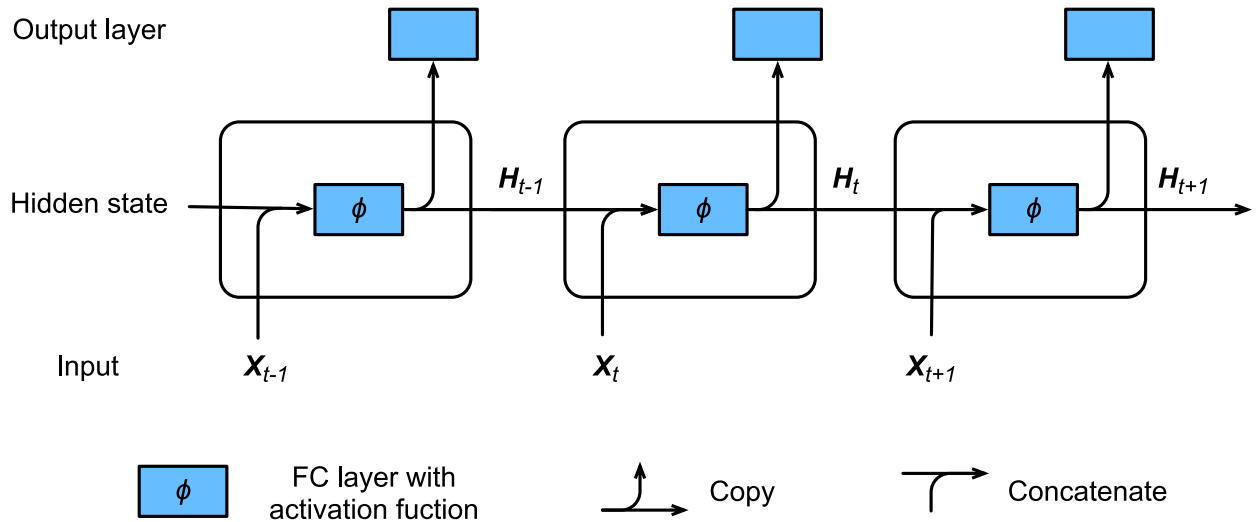
RNN: Nr parameters doesn't grow with sequence length

https://d2l.ai/chapter_recurrent-neural-networks/rnn.html#recurrent-networks-with-hidden-states

Recurrent Neural Network:
Process a sequence in order and share the same parameters at each time step.

For a sequence $\mathbf{X_t}$ for $t = 1, 2, 3, \ldots, T$

$$\mathbf{H_t} = \phi(\mathbf{X_t W_{xh}} + \mathbf{H_{t-1} W_{hh}} + \mathbf{b_h}), \qquad \text{output } \mathbf{O_t} = \mathbf{H_t W_{hq}} + \mathbf{b_q}$$



Q: How does the "concatenate" operator relate to the equation?
A: It's the same. $A \cdot B + C \cdot D = A|C \cdot \frac{B}{D}$ (where hor/ver lines are concatenations), see also last slide of this lecture.

Questions?

Gradients in RNNs

https://d2l.ai/chapter_recurrent-neural-networks/bptt.html

RNNs apply the same (shared weights) function at each time step:

$$z = f_3\left(\mathbf{x}_3, f_2\left(\mathbf{x}_2, f_1\left(\mathbf{x}_1, w\right), w\right), w\right)$$

Gradients in RNNs

https://d2l.ai/chapter_recurrent-neural-networks/bptt.html

RNNs apply the same (shared weights) function at each time step:

$$z = f_3\left(\mathbf{x}_3, f_2\left(\mathbf{x}_2, f_1\left(\mathbf{x}_1, w\right), w\right), w\right)$$

Length 3 recurrence derivative $\frac{dz}{dw}$ includes: $\frac{dz}{dw} = \frac{df_3}{df_2}\frac{df_2}{df_1}\frac{df_1}{dw}$

Gradients in RNNs

https://d2l.ai/chapter_recurrent-neural-networks/bptt.html

RNNs apply the same (shared weights) function at each time step:

$$z = f_3\left(\mathbf{x}_3, f_2\left(\mathbf{x}_2, f_1\left(\mathbf{x}_1, w\right), w\right), w\right)$$

Length 3 recurrence derivative $\frac{dz}{dw}$ includes: $\frac{dz}{dw} = \frac{df_3}{df_2}\frac{df_2}{df_1}\frac{df_1}{dw}$

Q: How about a recurrence of length T? A:

Gradients in RNNs

https://d2l.ai/chapter_recurrent-neural-networks/bptt.html

RNNs apply the same (shared weights) function at each time step:

$$z = f_3\left(\mathbf{x}_3, f_2\left(\mathbf{x}_2, f_1\left(\mathbf{x}_1, w\right), w\right), w\right)$$

Length 3 recurrence derivative $\frac{dz}{dw}$ includes: $\frac{dz}{dw} = \frac{df_3}{df_2}\frac{df_2}{df_1}\frac{df_1}{dw}$

Q: How about a recurrence of length T? A: $\frac{dz}{dw} = \frac{df_1}{dw}\prod_{t=2}^{T}\frac{df_t}{df_{t-1}}$

Gradients in RNNs

https://d2l.ai/chapter_recurrent-neural-networks/bptt.html

RNNs apply the same (shared weights) function at each time step:

$$z = f_3\left(\mathbf{x}_3, f_2\left(\mathbf{x}_2, f_1\left(\mathbf{x}_1, w\right), w\right), w\right)$$

Length 3 recurrence derivative $\frac{dz}{dw}$ includes: $\frac{dz}{dw} = \frac{df_3}{df_2}\frac{df_2}{df_1}\frac{df_1}{dw}$

Q: How about a recurrence of length T? A: $\frac{dz}{dw} = \frac{df_1}{dw}\prod_{t=2}^{T}\frac{df_t}{df_{t-1}}$

Q: If $|\frac{df_t}{df_{t-1}}| < 1$, what happens for $T = 1000$? A:

Gradients in RNNs

https://d2l.ai/chapter_recurrent-neural-networks/bptt.html

RNNs apply the same (shared weights) function at each time step:

$$z = f_3\left(\mathbf{x}_3, f_2\left(\mathbf{x}_2, f_1\left(\mathbf{x}_1, w\right), w\right), w\right)$$

Length 3 recurrence derivative $\frac{dz}{dw}$ includes: $\frac{dz}{dw} = \frac{df_3}{df_2}\frac{df_2}{df_1}\frac{df_1}{dw}$

Q: How about a recurrence of length T? A: $\frac{dz}{dw} = \frac{df_1}{dw}\prod_{t=2}^{T}\frac{df_t}{df_{t-1}}$

Q: If $|\frac{df_t}{df_{t-1}}| < 1$, what happens for $T = 1000$? A: $\frac{dz}{dw} \approx 0$ (vanishes)

Gradients in RNNs

https://d2l.ai/chapter_recurrent-neural-networks/bptt.html

RNNs apply the same (shared weights) function at each time step:

$$z = f_3\left(\mathbf{x}_3, f_2\left(\mathbf{x}_2, f_1\left(\mathbf{x}_1, w\right), w\right), w\right)$$

Length 3 recurrence derivative $\frac{dz}{dw}$ includes: $\frac{dz}{dw} = \frac{df_3}{df_2}\frac{df_2}{df_1}\frac{df_1}{dw}$

Q: How about a recurrence of length T? A: $\frac{dz}{dw} = \frac{df_1}{dw}\prod_{t=2}^{T}\frac{df_t}{df_{t-1}}$

Q: If $|\frac{df_t}{df_{t-1}}| < 1$, what happens for $T = 1000$? A: $\frac{dz}{dw} \approx 0$ (vanishes)

Q: If $|\frac{df_t}{df_{t-1}}| > 1$, what happens for $T = 1000$? A:

Gradients in RNNs

https://d2l.ai/chapter_recurrent-neural-networks/bptt.html

RNNs apply the same (shared weights) function at each time step:

$$z = f_3\left(\mathbf{x}_3, f_2\left(\mathbf{x}_2, f_1\left(\mathbf{x}_1, w\right), w\right), w\right)$$

Length 3 recurrence derivative $\frac{dz}{dw}$ includes: $\frac{dz}{dw} = \frac{df_3}{df_2}\frac{df_2}{df_1}\frac{df_1}{dw}$

Q: How about a recurrence of length T? A: $\frac{dz}{dw} = \frac{df_1}{dw}\prod_{t=2}^{T}\frac{df_t}{df_{t-1}}$

Q: If $|\frac{df_t}{df_{t-1}}| < 1$, what happens for $T = 1000$? A: $\frac{dz}{dw} \approx 0$ (vanishes)

Q: If $|\frac{df_t}{df_{t-1}}| > 1$, what happens for $T = 1000$? A: $\frac{dz}{dw} \approx \infty$ (explodes)

Gradients in RNNs

https://d2l.ai/chapter_recurrent-neural-networks/bptt.html

RNNs apply the same (shared weights) function at each time step:

$$z = f_3\left(\mathbf{x}_3, f_2\left(\mathbf{x}_2, f_1\left(\mathbf{x}_1, w\right), w\right), w\right)$$

Length 3 recurrence derivative $\frac{dz}{dw}$ includes: $\frac{dz}{dw} = \frac{df_3}{df_2}\frac{df_2}{df_1}\frac{df_1}{dw}$

Q: How about a recurrence of length T? A: $\frac{dz}{dw} = \frac{df_1}{dw}\prod_{t=2}^{T}\frac{df_t}{df_{t-1}}$

Q: If $|\frac{df_t}{df_{t-1}}| < 1$, what happens for $T = 1000$? A: $\frac{dz}{dw} \approx 0$ (vanishes)

Q: If $|\frac{df_t}{df_{t-1}}| > 1$, what happens for $T = 1000$? A: $\frac{dz}{dw} \approx \infty$ (explodes)

*The student[**s**], after reading the paper.. book.. blog.. was [**were**] well prepared.*

Gradients in RNNs

https://d2l.ai/chapter_recurrent-neural-networks/bptt.html

RNNs apply the same (shared weights) function at each time step:

$$z = f_3\left(\mathbf{x}_3, f_2\left(\mathbf{x}_2, f_1\left(\mathbf{x}_1, w\right), w\right), w\right)$$

Length 3 recurrence derivative $\frac{dz}{dw}$ includes: $\frac{dz}{dw} = \frac{df_3}{df_2}\frac{df_2}{df_1}\frac{df_1}{dw}$

Q: How about a recurrence of length T? A: $\frac{dz}{dw} = \frac{df_1}{dw}\prod_{t=2}^{T}\frac{df_t}{df_{t-1}}$

Q: If $|\frac{df_t}{df_{t-1}}| < 1$, what happens for $T = 1000$? A: $\frac{dz}{dw} \approx 0$ (vanishes)

Q: If $|\frac{df_t}{df_{t-1}}| > 1$, what happens for $T = 1000$? A: $\frac{dz}{dw} \approx \infty$ (explodes)

*The student[**s**], after reading the paper.. book.. blog.. was [**were**] well prepared.*

Q: What is the problem for such a sequence?

Gradients in RNNs

https://d2l.ai/chapter_recurrent-neural-networks/bptt.html

RNNs apply the same (shared weights) function at each time step:

$$z = f_3\left(\mathbf{x}_3, f_2\left(\mathbf{x}_2, f_1\left(\mathbf{x}_1, w\right), w\right), w\right)$$

Length 3 recurrence derivative $\frac{dz}{dw}$ includes: $\frac{dz}{dw} = \frac{df_3}{df_2}\frac{df_2}{df_1}\frac{df_1}{dw}$

Q: How about a recurrence of length T? A: $\frac{dz}{dw} = \frac{df_1}{dw}\prod_{t=2}^{T}\frac{df_t}{df_{t-1}}$

Q: If $|\frac{df_t}{df_{t-1}}| < 1$, what happens for $T = 1000$? A: $\frac{dz}{dw} \approx 0$ (vanishes)

Q: If $|\frac{df_t}{df_{t-1}}| > 1$, what happens for $T = 1000$? A: $\frac{dz}{dw} \approx \infty$ (explodes)

*The student[**s**], after reading the paper.. book.. blog.. was [**were**] well prepared.*

Q: What is the problem for such a sequence?
A: RNNs have difficulty with backpropping long-range relations

GRU: Gated Recurrent Unit

https://d2l.ai/chapter_recurrent-modern/gru.html

Improve RNNs hidden state $\mathbf{H_t}$ for long-range relationship learning.

GRU: Gated Recurrent Unit

https://d2l.ai/chapter_recurrent-modern/gru.html

Improve RNNs hidden state $\mathbf{H_t}$ for long-range relationship learning.

- Reset for different logical chunks in input (e.g.: chapters in a book)

GRU: Gated Recurrent Unit

https://d2l.ai/chapter_recurrent-modern/gru.html

Improve RNNs hidden state $\mathbf{H_t}$ for long-range relationship learning.

- Reset for different logical chunks in input (e.g.: chapters in a book)
- Do not update for uninformative input (e.g.: html codings)

GRU: Gated Recurrent Unit

https://d2l.ai/chapter_recurrent-modern/gru.html

Improve RNNs hidden state $\mathbf{H_t}$ for long-range relationship learning.

- Reset for different logical chunks in input (e.g.: chapters in a book)
- Do not update for uninformative input (e.g.: html codings)

Gate: Vector with entries in $(0, 1)$ Inspired by logical gates

GRU: Gated Recurrent Unit

https://d2l.ai/chapter_recurrent-modern/gru.html

Improve RNNs hidden state $\mathbf{H_t}$ for long-range relationship learning.

- Reset for different logical chunks in input (e.g.: chapters in a book)
- Do not update for uninformative input (e.g.: html codings)

Gate: Vector with entries in $(0, 1)$ Inspired by logical gates

Reset gate $\mathbf{R_t}$:

- Q: How can a gate $\mathbf{R_t}$ cause to ignore previous memories $\mathbf{H_{t-1}}$?

GRU: Gated Recurrent Unit

https://d2l.ai/chapter_recurrent-modern/gru.html

Improve RNNs hidden state $\mathbf{H_t}$ for long-range relationship learning.

- Reset for different logical chunks in input (e.g.: chapters in a book)
- Do not update for uninformative input (e.g.: html codings)

Gate: Vector with entries in $(0, 1)$ Inspired by logical gates

Reset gate $\mathbf{R_t}$:

- Q: How can a gate $\mathbf{R_t}$ cause to ignore previous memories $\mathbf{H_{t-1}}$?
- A: Multiply the gate $\mathbf{R_t}$ element-wise $\odot$ with $\mathbf{H_{t-1}}$.

GRU: Gated Recurrent Unit

https://d2l.ai/chapter_recurrent-modern/gru.html

Improve RNNs hidden state $\mathbf{H_t}$ for long-range relationship learning.

- Reset for different logical chunks in input (e.g.: chapters in a book)
- Do not update for uninformative input (e.g.: html codings)

Gate: Vector with entries in $(0, 1)$ Inspired by logical gates

Reset gate $\mathbf{R_t}$:

- Q: How can a gate $\mathbf{R_t}$ cause to ignore previous memories $\mathbf{H_{t-1}}$?
- A: Multiply the gate $\mathbf{R_t}$ element-wise $\odot$ with $\mathbf{H_{t-1}}$.

Update gate $\mathbf{Z_t}$:

- Q: For uninformative input $\mathbf{X_t}$, how can a gate $\mathbf{Z_t}$ weigh between:
  (the previous hidden state $\mathbf{H_{t-1}}$) *vs* (an already updated hidden state $\tilde{\mathbf{H}}_\mathbf{t}$)?

GRU: Gated Recurrent Unit

https://d2l.ai/chapter_recurrent-modern/gru.html

Improve RNNs hidden state $\mathbf{H_t}$ for long-range relationship learning.

- Reset for different logical chunks in input (e.g.: chapters in a book)
- Do not update for uninformative input (e.g.: html codings)

Gate: Vector with entries in $(0, 1)$ Inspired by logical gates

Reset gate $\mathbf{R_t}$:

- Q: How can a gate $\mathbf{R_t}$ cause to ignore previous memories $\mathbf{H_{t-1}}$?
- A: Multiply the gate $\mathbf{R_t}$ element-wise $\odot$ with $\mathbf{H_{t-1}}$.

Update gate $\mathbf{Z_t}$:

- Q: For uninformative input $\mathbf{X_t}$, how can a gate $\mathbf{Z_t}$ weigh between: (the previous hidden state $\mathbf{H_{t-1}}$) *vs* (an already updated hidden state $\tilde{\mathbf{H}}_\mathbf{t}$)?
- A: $\mathbf{H_{t-1}} \odot \mathbf{Z_t} + (\mathbf{1} - \mathbf{Z_t}) \odot \tilde{\mathbf{H}}_\mathbf{t}$, where $\tilde{\mathbf{H}}_\mathbf{t}$ is the already updated state.

GRU: Gated Recurrent Unit

https://d2l.ai/chapter_recurrent-modern/gru.html

Improve RNNs hidden state $\mathbf{H_t}$ for long-range relationship learning.

- Reset for different logical chunks in input (e.g.: chapters in a book)
- Do not update for uninformative input (e.g.: html codings)

Gate: Vector with entries in $(0, 1)$ Inspired by logical gates

Reset gate $\mathbf{R_t}$:

- Q: How can a gate $\mathbf{R_t}$ cause to ignore previous memories $\mathbf{H_{t-1}}$?
- A: Multiply the gate $\mathbf{R_t}$ element-wise $\odot$ with $\mathbf{H_{t-1}}$.

Update gate $\mathbf{Z_t}$:

- Q: For uninformative input $\mathbf{X_t}$, how can a gate $\mathbf{Z_t}$ weigh between: (the previous hidden state $\mathbf{H_{t-1}}$) *vs* (an already updated hidden state $\tilde{\mathbf{H}}_\mathbf{t}$)?
- A: $\mathbf{H_{t-1}} \odot \mathbf{Z_t} + (\mathbf{1} - \mathbf{Z_t}) \odot \tilde{\mathbf{H}}_\mathbf{t}$, where $\tilde{\mathbf{H}}_\mathbf{t}$ is the already updated state.
  Q: What happens for a $\mathbf{Z_t}$ as a vector of ones: $(1, 1, \ldots, 1)^\top$?

GRU: Gated Recurrent Unit

https://d2l.ai/chapter_recurrent-modern/gru.html

Improve RNNs hidden state $\mathbf{H_t}$ for long-range relationship learning.

- Reset for different logical chunks in input (e.g.: chapters in a book)
- Do not update for uninformative input (e.g.: html codings)

Gate: Vector with entries in $(0, 1)$ Inspired by logical gates

Reset gate $\mathbf{R_t}$:

- Q: How can a gate $\mathbf{R_t}$ cause to ignore previous memories $\mathbf{H_{t-1}}$?
- A: Multiply the gate $\mathbf{R_t}$ element-wise $\odot$ with $\mathbf{H_{t-1}}$.

Update gate $\mathbf{Z_t}$:

- Q: For uninformative input $\mathbf{X_t}$, how can a gate $\mathbf{Z_t}$ weigh between: (the previous hidden state $\mathbf{H_{t-1}}$) *vs* (an already updated hidden state $\tilde{\mathbf{H}}_\mathbf{t}$)?
- A: $\mathbf{H_{t-1}} \odot \mathbf{Z_t} + (\mathbf{1} - \mathbf{Z_t}) \odot \tilde{\mathbf{H}}_\mathbf{t}$, where $\tilde{\mathbf{H}}_\mathbf{t}$ is the already updated state.
  Q: What happens for a $\mathbf{Z_t}$ as a vector of ones: $(1, 1, \ldots, 1)^\top$?
  A: Input $\mathbf{X_t}$ has no effect: only previous state $\mathbf{H_{t-1}}$ used (ie: no update).

GRU: Gated Recurrent Unit

https://d2l.ai/chapter_recurrent-modern/gru.html

Gates allow to selectively keep or remove dimensions from the hidden state.

GRU: Gated Recurrent Unit

https://d2l.ai/chapter_recurrent-modern/gru.html

Gates allow to selectively keep or remove dimensions from the hidden state.

Q: How to force gates $\mathbf{R_t}$ and $\mathbf{Z_t}$ to be vectors in $(0, 1)$?

GRU: Gated Recurrent Unit

https://d2l.ai/chapter_recurrent-modern/gru.html

Gates allow to selectively keep or remove dimensions from the hidden state.

Q: How to force gates $\mathbf{R_t}$ and $\mathbf{Z_t}$ to be vectors in $(0, 1)$?
A: Use sigmoid activation σ.

GRU: Gated Recurrent Unit

https://d2l.ai/chapter_recurrent-modern/gru.html

Gates allow to selectively keep or remove dimensions from the hidden state.

Q: How to force gates $\mathbf{R_t}$ and $\mathbf{Z_t}$ to be vectors in $(0, 1)$?
A: Use sigmoid activation σ.

Q: Based on what information would you learn the gates? (what info is available?)

GRU: Gated Recurrent Unit

https://d2l.ai/chapter_recurrent-modern/gru.html

Gates allow to selectively keep or remove dimensions from the hidden state.

Q: How to force gates $\mathbf{R_t}$ and $\mathbf{Z_t}$ to be vectors in $(0, 1)$?
A: Use sigmoid activation σ.

Q: Based on what information would you learn the gates? (what info is available?)
A: The current input $\mathbf{X_t}$ and previous state $\mathbf{H_{t-1}}$:

GRU: Gated Recurrent Unit

https://d2l.ai/chapter_recurrent-modern/gru.html

Gates allow to selectively keep or remove dimensions from the hidden state.

Q: How to force gates $\mathbf{R_t}$ and $\mathbf{Z_t}$ to be vectors in $(0, 1)$?
A: Use sigmoid activation σ.

Q: Based on what information would you learn the gates? (what info is available?)
A: The current input $\mathbf{X_t}$ and previous state $\mathbf{H_{t-1}}$:

$$\mathbf{R_t} = \sigma(\mathbf{X_t}\mathbf{W_{xr}} + \mathbf{H_{t-1}}\mathbf{W_{hr}} + \mathbf{b_r})$$
$$\mathbf{Z_t} = \sigma(\mathbf{X_t}\mathbf{W_{xz}} + \mathbf{H_{t-1}}\mathbf{W_{hz}} + \mathbf{b_z})$$

GRU: Gated Recurrent Unit

https://d2l.ai/chapter_recurrent-modern/gru.html

Gates allow to selectively keep or remove dimensions from the hidden state.

Q: How to force gates $\mathbf{R_t}$ and $\mathbf{Z_t}$ to be vectors in $(0, 1)$?
A: Use sigmoid activation σ.

Q: Based on what information would you learn the gates? (what info is available?)
A: The current input $\mathbf{X_t}$ and previous state $\mathbf{H_{t-1}}$:

$$\mathbf{R_t} = \sigma(\mathbf{X_t}\mathbf{W_{xr}} + \mathbf{H_{t-1}}\mathbf{W_{hr}} + \mathbf{b_r})$$
$$\mathbf{Z_t} = \sigma(\mathbf{X_t}\mathbf{W_{xz}} + \mathbf{H_{t-1}}\mathbf{W_{hz}} + \mathbf{b_z})$$

Q: Looks familiar?

GRU: Gated Recurrent Unit

https://d2l.ai/chapter_recurrent-modern/gru.html

Gates allow to selectively keep or remove dimensions from the hidden state.

Q: How to force gates $\mathbf{R_t}$ and $\mathbf{Z_t}$ to be vectors in $(0, 1)$?
A: Use sigmoid activation σ.

Q: Based on what information would you learn the gates? (what info is available?)
A: The current input $\mathbf{X_t}$ and previous state $\mathbf{H_{t-1}}$:

$$\mathbf{R_t} = \sigma(\mathbf{X_t}\mathbf{W_{xr}} + \mathbf{H_{t-1}}\mathbf{W_{hr}} + \mathbf{b_r})$$
$$\mathbf{Z_t} = \sigma(\mathbf{X_t}\mathbf{W_{xz}} + \mathbf{H_{t-1}}\mathbf{W_{hz}} + \mathbf{b_z})$$

Q: Looks familiar? A: Standard RNN: $\mathbf{H_t} = \phi(\mathbf{X_t}\mathbf{W_{xh}} + \mathbf{H_{t-1}}\mathbf{W_{hh}} + \mathbf{b_h})$

GRU: Gated Recurrent Unit

https://d2l.ai/chapter_recurrent-modern/gru.html

Gates allow to selectively keep or remove dimensions from the hidden state.

Q: How to force gates $\mathbf{R_t}$ and $\mathbf{Z_t}$ to be vectors in $(0, 1)$?
A: Use sigmoid activation σ.

Q: Based on what information would you learn the gates? (what info is available?)
A: The current input $\mathbf{X_t}$ and previous state $\mathbf{H_{t-1}}$:

$$\mathbf{R_t} = \sigma(\mathbf{X_t}\mathbf{W_{xr}} + \mathbf{H_{t-1}}\mathbf{W_{hr}} + \mathbf{b_r})$$
$$\mathbf{Z_t} = \sigma(\mathbf{X_t}\mathbf{W_{xz}} + \mathbf{H_{t-1}}\mathbf{W_{hz}} + \mathbf{b_z})$$

Q: Looks familiar? A: Standard RNN: $\mathbf{H_t} = \phi(\mathbf{X_t}\mathbf{W_{xh}} + \mathbf{H_{t-1}}\mathbf{W_{hh}} + \mathbf{b_h})$

How a GRU differs from a standard RNN:

GRU adds a reset gate:

- Candidate state $\tilde{\mathbf{H}}_\mathbf{t} = \tanh(\mathbf{X_t}\mathbf{W_{xh}} + (\mathbf{R_t} \odot \mathbf{H_{t-1}})\mathbf{W_{hh}} + \mathbf{b_h})$

GRU: Gated Recurrent Unit

https://d2l.ai/chapter_recurrent-modern/gru.html

Gates allow to selectively keep or remove dimensions from the hidden state.

Q: How to force gates $\mathbf{R_t}$ and $\mathbf{Z_t}$ to be vectors in $(0, 1)$?
A: Use sigmoid activation σ.

Q: Based on what information would you learn the gates? (what info is available?)
A: The current input $\mathbf{X_t}$ and previous state $\mathbf{H_{t-1}}$:

$$\mathbf{R_t} = \sigma(\mathbf{X_t W_{xr}} + \mathbf{H_{t-1} W_{hr}} + \mathbf{b_r})$$
$$\mathbf{Z_t} = \sigma(\mathbf{X_t W_{xz}} + \mathbf{H_{t-1} W_{hz}} + \mathbf{b_z})$$

Q: Looks familiar? A: Standard RNN: $\mathbf{H_t} = \phi(\mathbf{X_t W_{xh}} + \mathbf{H_{t-1} W_{hh}} + \mathbf{b_h})$

How a GRU differs from a standard RNN:

GRU adds a reset gate:

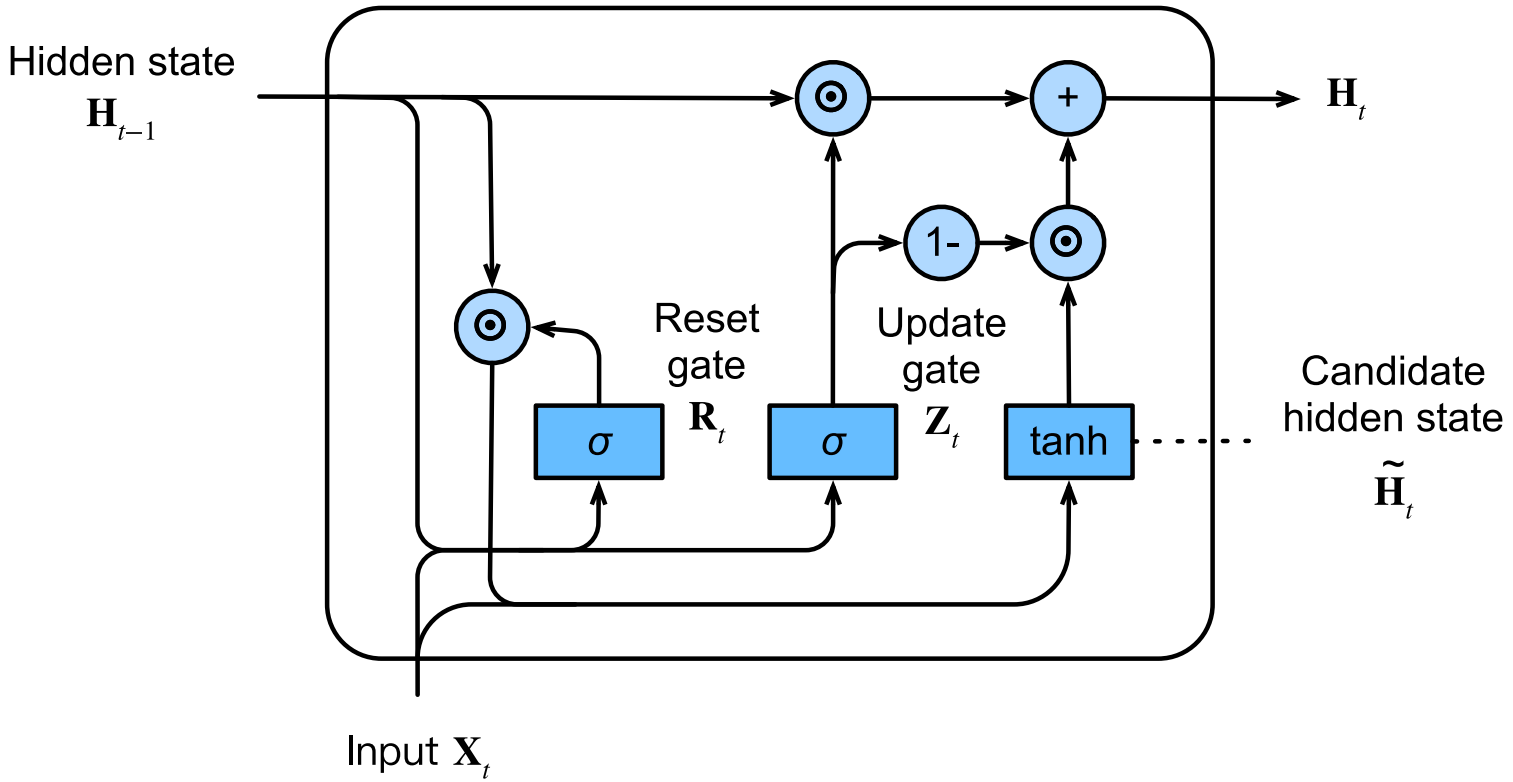
- Candidate state $\tilde{\mathbf{H}}_\mathbf{t} = \tanh(\mathbf{X_t W_{xh}} + (\mathbf{R_t} \odot \mathbf{H_{t-1}})\mathbf{W_{hh}} + \mathbf{b_h})$

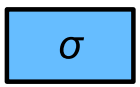
GRU adds an update gate:

- Current state $\mathbf{H_t} = \mathbf{H_{t-1}} \odot \mathbf{Z_t} + (\mathbf{1} - \mathbf{Z_t}) \odot \tilde{\mathbf{H}}_\mathbf{t}$, where $\tilde{\mathbf{H}}_\mathbf{t}$ is candidate state.

GRU: Gated Recurrent Unit

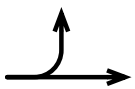
https://d2l.ai/chapter_recurrent-modern/gru.html



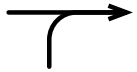


FC layer with activation fuction

Elementwise operator



Copy



Concatenate

Questions?

Questions?

Q: Why does gating allow for long-term relations? (ie: what about vanishing/exploding gradients?)

Questions?

Q: Why does gating allow for long-term relations? (ie: what about vanishing/exploding gradients?)
A: Relevant long term information can pass through unchanged.

LSTM: Long Short Term Memory networks

https://d2l.ai/chapter_recurrent-modern/lstm.html
See also: https://colah.github.io/posts/2015-08-Understanding-LSTMs

Input gate, Forget gate, Output gate, and an additional 'memory cell' state $\mathbf{C_t}$

LSTM: Long Short Term Memory networks

https://d2l.ai/chapter_recurrent-modern/lstm.html
See also: https://colah.github.io/posts/2015-08-Understanding-LSTMs

Input gate, Forget gate, Output gate, and an additional 'memory cell' state $\mathbf{C_t}$

- LSTM linearly adds/removes information to 'conveyor belt' cell $\mathbf{C_t}$

LSTM: Long Short Term Memory networks

https://d2l.ai/chapter_recurrent-modern/lstm.html
See also: https://colah.github.io/posts/2015-08-Understanding-LSTMs

Input gate, Forget gate, Output gate, and an additional 'memory cell' state $\mathbf{C_t}$

- LSTM linearly adds/removes information to 'conveyor belt' cell $\mathbf{C_t}$
- Gates and candidate Cell depend on input and previous hidden state.

LSTM: Long Short Term Memory networks

https://d2l.ai/chapter_recurrent-modern/lstm.html
See also: https://colah.github.io/posts/2015-08-Understanding-LSTMs

Input gate, Forget gate, Output gate, and an additional 'memory cell' state $\mathbf{C_t}$

- LSTM linearly adds/removes information to 'conveyor belt' cell $\mathbf{C_t}$
- Gates and candidate Cell depend on input and previous hidden state.
- Current Memory cell depends on previous Cell and candidate Cell

LSTM: Long Short Term Memory networks

https://d2l.ai/chapter_recurrent-modern/lstm.html
See also: https://colah.github.io/posts/2015-08-Understanding-LSTMs

Input gate, Forget gate, Output gate, and an additional 'memory cell' state $\mathbf{C_t}$

- LSTM linearly adds/removes information to 'conveyor belt' cell $\mathbf{C_t}$
- Gates and candidate Cell depend on input and previous hidden state.
- Current Memory cell depends on previous Cell and candidate Cell
- Current hidden state depends on current Cell

LSTM: Long Short Term Memory networks

https://d2l.ai/chapter_recurrent-modern/lstm.html
See also: https://colah.github.io/posts/2015-08-Understanding-LSTMs

Input gate, Forget gate, Output gate, and an additional 'memory cell' state $\mathbf{C_t}$

- LSTM linearly adds/removes information to 'conveyor belt' cell $\mathbf{C_t}$
- Gates and candidate Cell depend on input and previous hidden state.
- Current Memory cell depends on previous Cell and candidate Cell
- Current hidden state depends on current Cell

Gates computed like in a GRU:

LSTM: Long Short Term Memory networks

https://d2l.ai/chapter_recurrent-modern/lstm.html
See also: https://colah.github.io/posts/2015-08-Understanding-LSTMs

Input gate, Forget gate, Output gate, and an additional 'memory cell' state $\mathbf{C_t}$

- LSTM linearly adds/removes information to 'conveyor belt' cell $\mathbf{C_t}$
- Gates and candidate Cell depend on input and previous hidden state.
- Current Memory cell depends on previous Cell and candidate Cell
- Current hidden state depends on current Cell

Gates computed like in a GRU:

Input gate	$\mathbf{I_t} = \sigma(\mathbf{X_t W_{xi}} + \mathbf{H_{t-1} W_{hi}} + \mathbf{b_i})$
Forget gate	$\mathbf{F_t} = \sigma(\mathbf{X_t W_{xf}} + \mathbf{H_{t-1} W_{hf}} + \mathbf{b_f})$
Output gate	$\mathbf{O_t} = \sigma(\mathbf{X_t W_{xo}} + \mathbf{H_{t-1} W_{ho}} + \mathbf{b_o})$

LSTM: Long Short Term Memory networks

https://d2l.ai/chapter_recurrent-modern/lstm.html
See also: https://colah.github.io/posts/2015-08-Understanding-LSTMs

Input gate, Forget gate, Output gate, and an additional 'memory cell' state $\mathbf{C_t}$

- LSTM linearly adds/removes information to 'conveyor belt' cell $\mathbf{C_t}$
- Gates and candidate Cell depend on input and previous hidden state.
- Current Memory cell depends on previous Cell and candidate Cell
- Current hidden state depends on current Cell

Gates computed like in a GRU:

Input gate	$\mathbf{I_t} = \sigma(\mathbf{X_t W_{xi}} + \mathbf{H_{t-1} W_{hi}} + \mathbf{b_i})$
Forget gate	$\mathbf{F_t} = \sigma(\mathbf{X_t W_{xf}} + \mathbf{H_{t-1} W_{hf}} + \mathbf{b_f})$
Output gate	$\mathbf{O_t} = \sigma(\mathbf{X_t W_{xo}} + \mathbf{H_{t-1} W_{ho}} + \mathbf{b_o})$

Candidate memory cell $\tilde{\mathbf{C}}_\mathbf{t} = \tanh(\mathbf{X_t W_{xc}} + \mathbf{H_{t-1} W_{hc}} + \mathbf{b_c})$

LSTM: Long Short Term Memory networks

https://d2l.ai/chapter_recurrent-modern/lstm.html
See also: https://colah.github.io/posts/2015-08-Understanding-LSTMs

Input gate, Forget gate, Output gate, and an additional 'memory cell' state $\mathbf{C_t}$

- LSTM linearly adds/removes information to 'conveyor belt' cell $\mathbf{C_t}$
- Gates and candidate Cell depend on input and previous hidden state.
- Current Memory cell depends on previous Cell and candidate Cell
- Current hidden state depends on current Cell

Gates computed like in a GRU:

Input gate $\mathbf{I_t} = \sigma(\mathbf{X_t W_{xi}} + \mathbf{H_{t-1} W_{hi}} + \mathbf{b_i})$
Forget gate $\mathbf{F_t} = \sigma(\mathbf{X_t W_{xf}} + \mathbf{H_{t-1} W_{hf}} + \mathbf{b_f})$
Output gate $\mathbf{O_t} = \sigma(\mathbf{X_t W_{xo}} + \mathbf{H_{t-1} W_{ho}} + \mathbf{b_o})$

Candidate memory cell $\tilde{\mathbf{C}}_\mathbf{t} = \tanh(\mathbf{X_t W_{xc}} + \mathbf{H_{t-1} W_{hc}} + \mathbf{b_c})$

Where GRU has a single update gate: $\mathbf{H_t} = \mathbf{H_{t-1}} \odot \mathbf{Z_t} + (\mathbf{1} - \mathbf{Z_t}) \odot \tilde{\mathbf{H}}_\mathbf{t}$)

LSTM has 2 gates: Forget gate $\mathbf{F_t}$; input gate $\mathbf{I_t}$: $\mathbf{C_t} = \mathbf{F_t} \odot \mathbf{C_{t-1}} + \mathbf{I_t} \odot \tilde{\mathbf{C}}_\mathbf{t}$

LSTM: Long Short Term Memory networks

https://d2l.ai/chapter_recurrent-modern/lstm.html
See also: https://colah.github.io/posts/2015-08-Understanding-LSTMs

Input gate, Forget gate, Output gate, and an additional 'memory cell' state $\mathbf{C_t}$

- LSTM linearly adds/removes information to 'conveyor belt' cell $\mathbf{C_t}$
- Gates and candidate Cell depend on input and previous hidden state.
- Current Memory cell depends on previous Cell and candidate Cell
- Current hidden state depends on current Cell

Gates computed like in a GRU:

Input gate	$\mathbf{I_t} = \sigma(\mathbf{X_t W_{xi}} + \mathbf{H_{t-1} W_{hi}} + \mathbf{b_i})$
Forget gate	$\mathbf{F_t} = \sigma(\mathbf{X_t W_{xf}} + \mathbf{H_{t-1} W_{hf}} + \mathbf{b_f})$
Output gate	$\mathbf{O_t} = \sigma(\mathbf{X_t W_{xo}} + \mathbf{H_{t-1} W_{ho}} + \mathbf{b_o})$

Candidate memory cell $\tilde{\mathbf{C}}_\mathbf{t} = \tanh(\mathbf{X_t W_{xc}} + \mathbf{H_{t-1} W_{hc}} + \mathbf{b_c})$

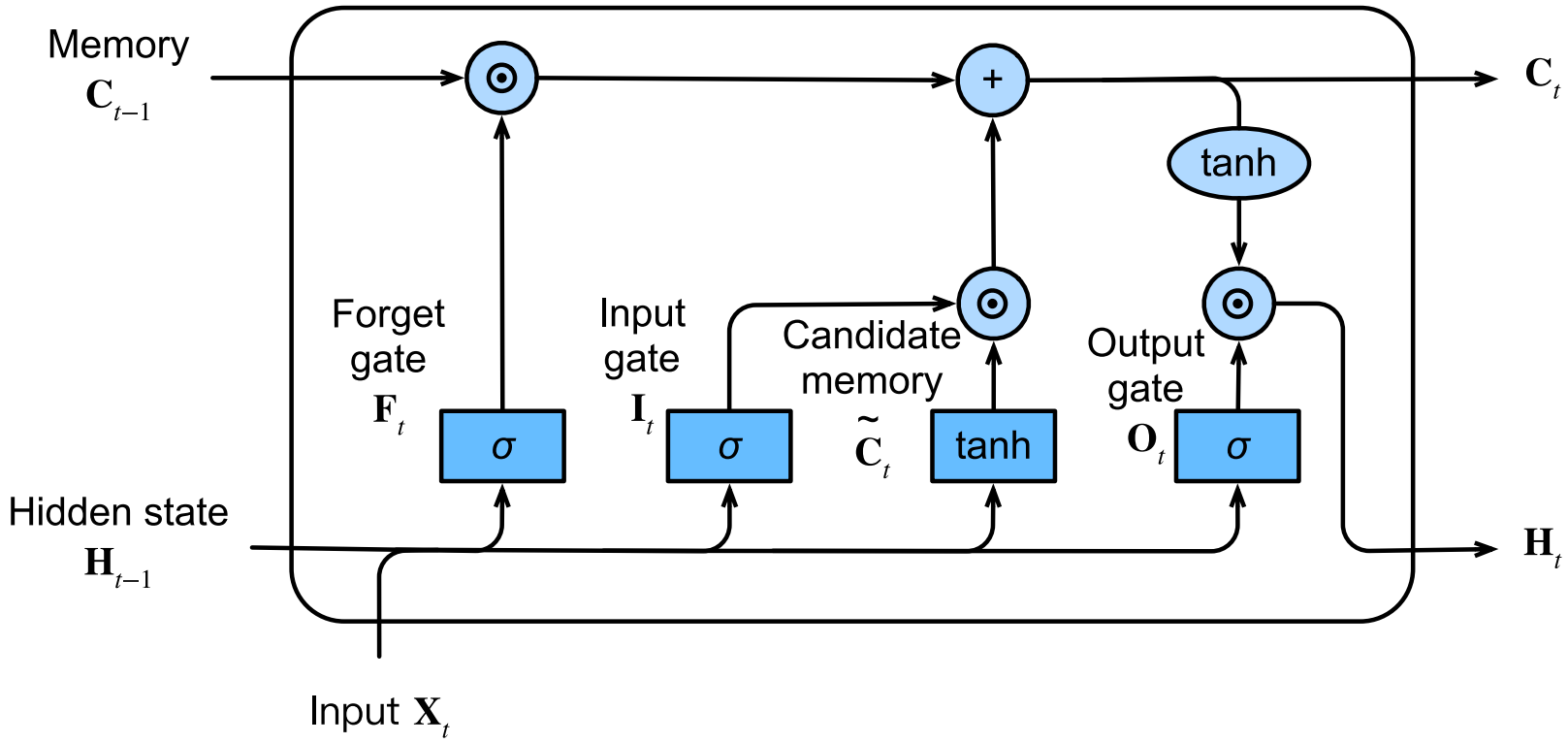
Where GRU has a single update gate: $\mathbf{H_t} = \mathbf{H_{t-1}} \odot \mathbf{Z_t} + (\mathbf{1} - \mathbf{Z_t}) \odot \tilde{\mathbf{H}}_\mathbf{t}$)

LSTM has 2 gates: Forget gate $\mathbf{F_t}$; input gate $\mathbf{I_t}$: $\mathbf{C_t} = \mathbf{F_t} \odot \mathbf{C_{t-1}} + \mathbf{I_t} \odot \tilde{\mathbf{C}}_\mathbf{t}$

Current hidden state uses output gate: $\mathbf{H_t} = \mathbf{O_t} \odot \tanh(\mathbf{C_t})$

LSTM: Long Short Term Memory

D2L-book chapter 9.2



σ FC layer with activation fuction

Elementwise operator

Copy

Concatenate

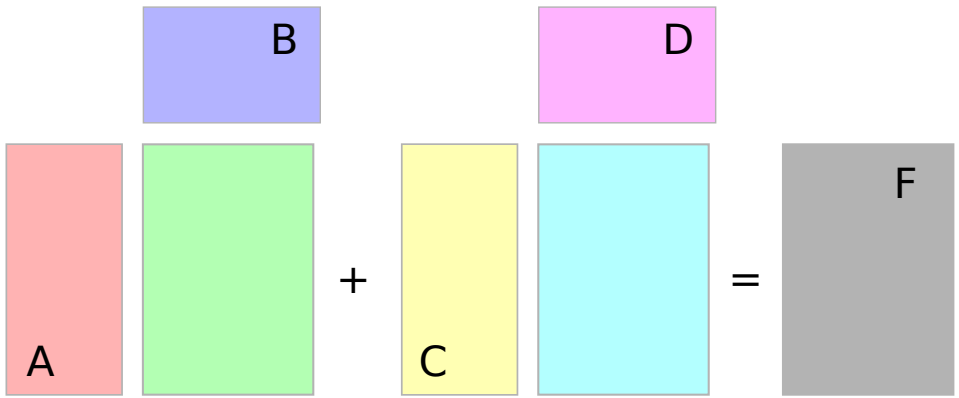
Questions?

Summary

- RNNs
- Vanishing/exploding gradients in RNNs
- GRU and LSTM

Appendix: Concatenating matrices

Adding



Concatenation