11-695: AI Engineering
Recurrent Neural Networks

LTI/SCS

Spring 2020

Outline

Carnegie Mellon

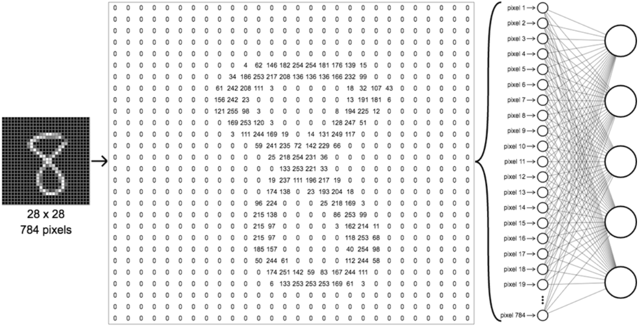
1. Motivation
2. RNNs as Functions Composition
3. Modelling RNNs
4. Extensions

How to “model”each type of data?

- Spatial: images
- Sequential: text
- Mixed: videos
- Other structured and unstructured data: graphs, graphical models, ...

Image credit: BYU Art

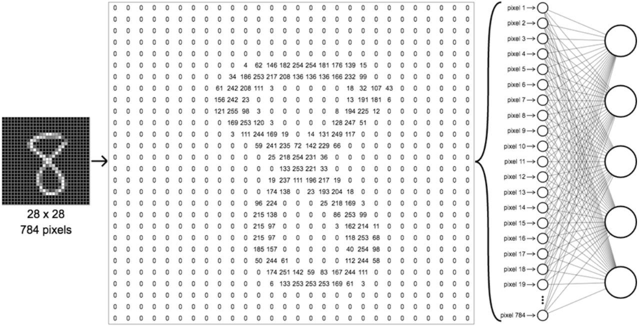
MLP a one-size-fits-all approximator?



- A general solution to general data

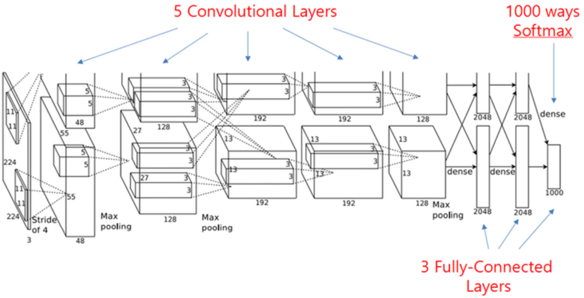
MLP a one-size-fits-all approximator?

Carnegie Mellon



- A general solution to general data
- It has disadvantages: params, relations, ...

How about CNN in modelling data?

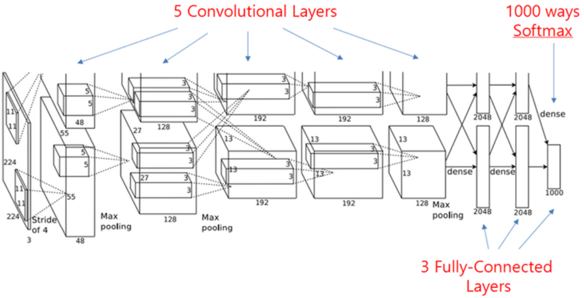


- It has a huge advantage in vision and related data

[1] Not counting some special cases such as Fully CNN

Image credit: Krizhevsky et. al 2012

How about CNN in modelling data?

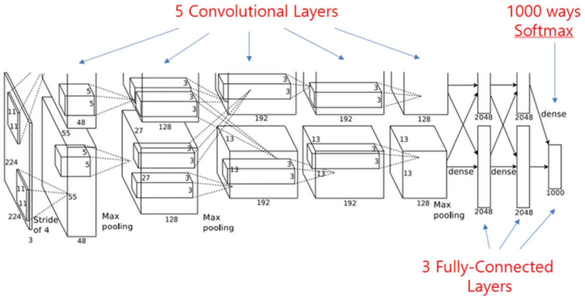


- It has a huge advantage in vision and related data
- But also normally requires fixed size of input[1]

[1] Not counting some special cases such as Fully CNN

Image credit: Krizhevsky et. al 2012

How about CNN in modelling data?

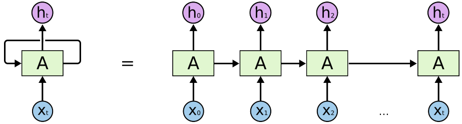


- It has a huge advantage in vision and related data
- But also normally requires fixed size of input[1]
- With MLP, they are feed-foward type without "memory"

[1] Not counting some special cases such as Fully CNN

Image credit: Krizhevsky et. al 2012

Model sequential data

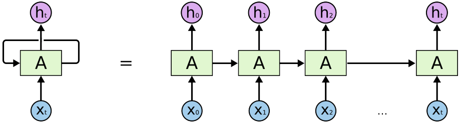


- Texts, video, trading, ...
- Treating samples indepdendently is losing information
- We need a special model for time-series data
- Markov, Graphical Model, ... and

Image credit: Chris Olah

Model sequential data

CarnegieMellon



- Texts, video, trading, ...
- Treating samples indepdendently is losing information
- We need a special model for time-series data
- Markov, Graphical Model, ... and
- Recurrent NN

Image credit: Chris Olah

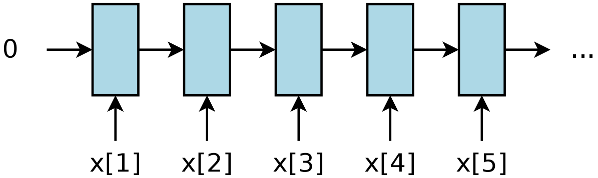
Outline

Carnegie Mellon

1. Motivation
2. RNNs as Functions Composition
3. Modelling RNNs
4. Extensions

Recurrent Neural Networks

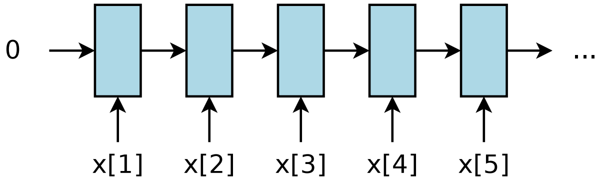
Carnegie Mellon



- Processes a sequence of signals
  - $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T \in \mathbb{R}^D$
- ... in a sequential order
  - $\mathbf{h}_0 = \mathbf{0}_H$
  - $\mathbf{h}_t = \mathbf{f}(\mathbf{x}_t, \mathbf{h}_{t-1})$
- Designing an RNN primarily means modelling sequential representation with a function $\mathbf{f} : \mathbb{R}^D \times \mathbb{R}^H \rightarrow \mathbb{R}^H$

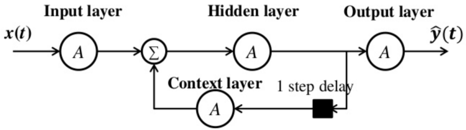
Example 1: A Simplistic RNN

Carnegie Mellon



- With the function $\mathbf{f}(\mathbf{x}, \mathbf{h}) = \mathbf{x} + \mathbf{h}$
  - $\mathbf{h}_0 = \mathbf{0}_H$
  - $\mathbf{h}_t = \mathbf{h}_{t-1} + \mathbf{x}_t$
- This network requires $D = H$, and is really naive ...

Example 2: Vanilla RNN (Ellman)[2]

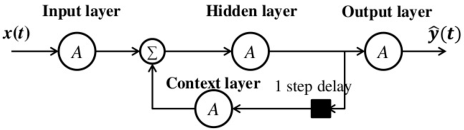


- With the function $\mathbf{f}(\mathbf{x}, \mathbf{h}) = g(\mathbf{x} \cdot \mathbf{W}_{ih} + \mathbf{h} \cdot \mathbf{W}_{hh} + \mathbf{b}_{hh})$
  - g is an activation function, e.g. tanh, sigmoid, ...
  - $\mathbf{W}_{ih} \in \mathbb{R}^{D \times H}$, $\mathbf{W}_{hh} \in \mathbb{R}^{H \times H}$ are the *shared* parameters.
- Much efficient, invented in 1990, got popular in 2011.

[2] https://onlinelibrary.wiley.com/doi/epdf/10.1207/s15516709cog1402_1

Image credit: Viacheslav Khomenko

Example 2: Vanilla RNN (Ellman)[2]

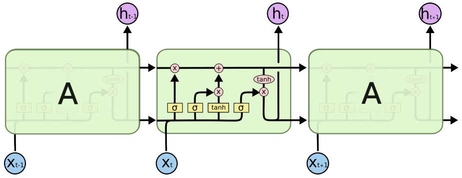


- With the function $\mathbf{f}(\mathbf{x}, \mathbf{h}) = g(\mathbf{x} \cdot \mathbf{W}_{ih} + \mathbf{h} \cdot \mathbf{W}_{hh} + \mathbf{b}_{hh})$
  - g is an activation function, e.g. tanh, sigmoid, ...
  - $\mathbf{W}_{ih} \in \mathbb{R}^{D \times H}$, $\mathbf{W}_{hh} \in \mathbb{R}^{H \times H}$ are the *shared* parameters.
- Much efficient, invented in 1990, got popular in 2011.
- Vanishing gradients problem

[2] https://onlinelibrary.wiley.com/doi/epdf/10.1207/s15516709cog1402_1

Image credit: Viacheslav Khomenko

Example 3: Long Short-Term Memory (LSTM)[3]



- Invented in 1997 and got super popular since 2014.

$$\begin{bmatrix} \mathbf{i} \\ \mathbf{f} \\ \mathbf{o} \\ \mathbf{g} \end{bmatrix}^\top = \begin{bmatrix} \text{sigmoid} \\ \tanh \\ \text{sigmoid} \\ \text{sigmoid} \end{bmatrix} \mathbf{W}_{H \times (D+H)} \cdot \begin{bmatrix} \mathbf{x}_t^\top \\ \mathbf{h}_t^\top \end{bmatrix} \tag{1}$$

$$\mathbf{c}_t = \mathbf{i} \otimes \mathbf{g} + \mathbf{f} \cdot \mathbf{c}_{t-1}$$

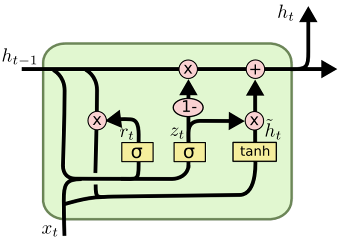
$$\mathbf{h}_t = \mathbf{o} \otimes \tanh \mathbf{c}_t$$

[3] https://www.bioinf.jku.at/publications/older/2604.pdf

Image credit: Chris Olah

Example 4: Gated Recurrent Units (GRU)[4]

CarnegieMellon



- Combines forget and input gates to change **f**

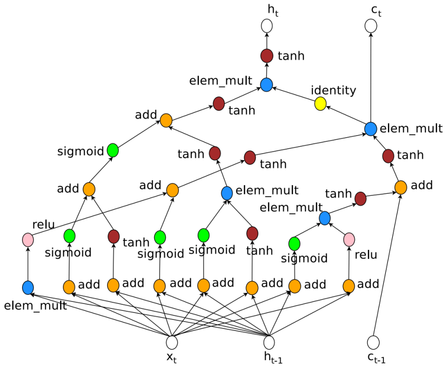
$$
\begin{aligned}
\mathbf{z} &= \mathrm{sigmoid}(\mathbf{x}_t \cdot \mathbf{W}_{xz} + \mathbf{h}_{t-1} \cdot \mathbf{W}_{hz}) \\
\mathbf{r} &= \mathrm{sigmoid}(\mathbf{x}_t \cdot \mathbf{W}_{xr} + \mathbf{h}_{t-1} \cdot \mathbf{W}_{hr}) \\
\tilde{\mathbf{h}} &= \mathrm{sigmoid}(\mathbf{x}_t \cdot \tilde{\mathbf{W}}_{x} + (\mathbf{r} \cdot \mathbf{h}_{t-1}) \cdot \tilde{\mathbf{W}}_{h}) \\
\mathbf{h}_t &= (1 - \mathbf{z}) \otimes \mathbf{h}_{t-1} + \mathbf{z} \otimes \tilde{\mathbf{h}}
\end{aligned}
\tag{2}
$$

- Cell state is same as hidden's (output), unlike LSTM, so simpler

[4] https://arxiv.org/pdf/1409.1259.pdf

Image credit: Chris Olah

Example 5: Neural Architecture Search[5]

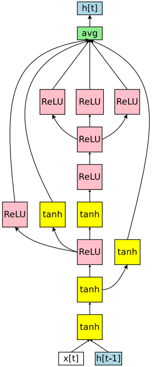


- You can also use a computer to generate good formulas for **f**

[5] https://arxiv.org/pdf/1611.01578.pdf

Image credit: Barret Zoph and Quoc Le

Example 6: Efficient Neural Architecture Search[6]



- Yet another one, also generated by a computer

[6] https://arxiv.org/pdf/1802.03268.pdf

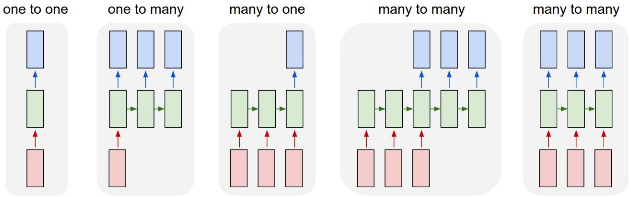
Image credit: Hieu Pham *et al.*

Outline

1. Motivation
2. RNNs as Functions Composition
3. Modelling RNNs
4. Extensions

RNN Model Variations

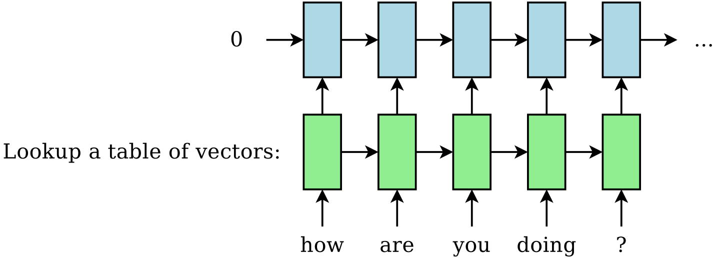
Carnegie Mellon



- You can be model many layouts of input and outputs
- You can be *very* creative about RNNs:
  - How to choose the input sequence $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T$
  - What to do with the "hidden" sequence $\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_T$

Image credit: Andrej Karpathy

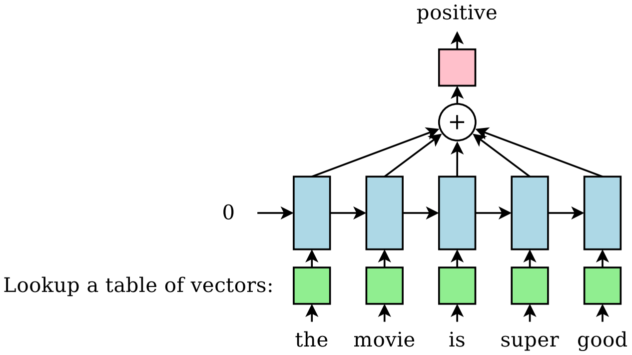
Flexible Input: Word embeddings[7]



- To process a sequence of words
  - Store a dictionary that maps words to vectors in $\mathbf{R}^D$
  - Use these $\mathbb{R}^D$ vectors as inputs to an RNN.
- Each word is embedded to a lower dimensional space

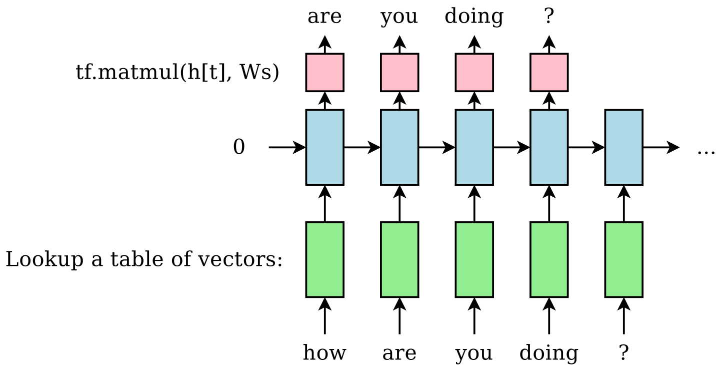
[7] http://jmlr.org/papers/volume3/bengio03a/bengio03a.pdf

Output: Adding a Softmax head



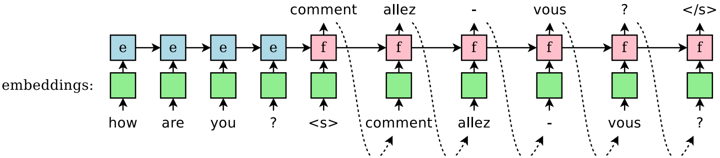
- You can sum over the $\mathbf{h}_t$ and hook up a softmax head to make a prediction about your sequence
  - Example: sentiment analysis

Output: Multiple softmax heads



- You can use the $\mathbf{h}_t$ as to predict the next word in your sequence
  - Example: *language model*
  - Because it can model $p(w_t|w_{<t})$

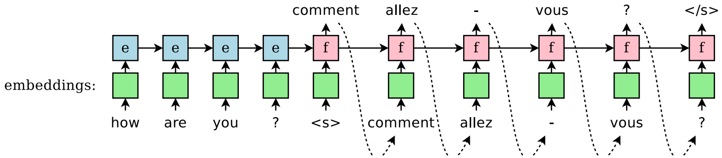
Output: Sequence-to-Sequence models[8]



- Only use the softmax heads where you want to generate a translated word
  - It's called *neural machine translation*
  - Because it can model $p(\mathbf{t}_t | \mathbf{t}_{<t}, \mathbf{s})$
- A **very important model** to remember!
- Can be extended to many other problems in practice such as image captioning, image synthesis, multimodal models, ...

[8] https://arxiv.org/pdf/1409.3215.pdf

Output: Attention[9]



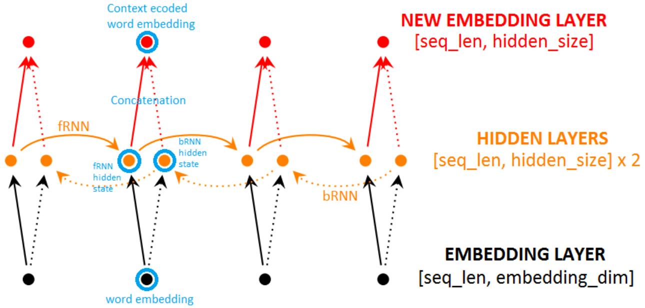
- Yet another way to manipulate your $\mathbf{h}_t$ states.
- $\mathbf{e}_i$, $\mathbf{f}_j$ are your blue and red states

$$
\begin{aligned}
\alpha_{j,i} &= g(\mathbf{f}_j, \mathbf{e}_i) \\
a_{j,i} &= \mathrm{Softmax}(\alpha_{j,1}, \alpha_{j,2}, ..., \alpha_{j,|\mathbf{s}|}) \\
c_j &= \sum_{i=1}^{|\mathbf{s}|} a_{j,i}\mathbf{e}_i
\end{aligned} \tag{3}
$$

[9] https://arxiv.org/pdf/1409.0473.pdf

Outline

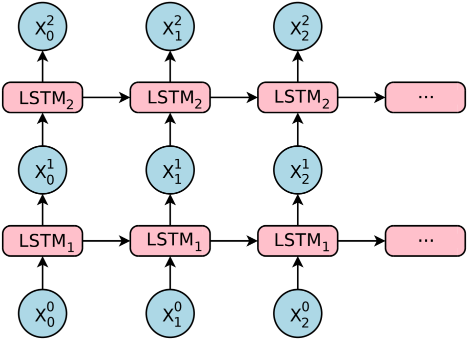
Bidirectional RNNs



- Model 2-way relations between time steps: past & future
- Actually contain 2 RNNs with opposite directions
- Usually concatenate the hidden states of the two

Image credit: Chaitanya Joshi

Deep RNNs

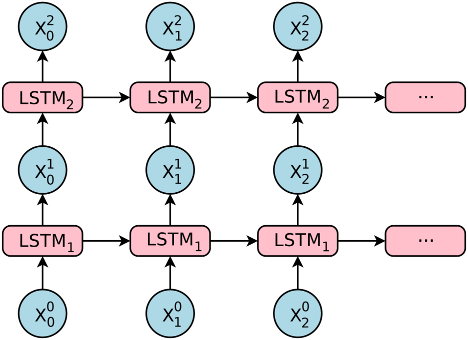


- Simply stacking layers on top of each other

[10] https://arxiv.org/pdf/1409.3215.pdf

Image credit: Yonghui Wu *et al.*

Deep RNNs

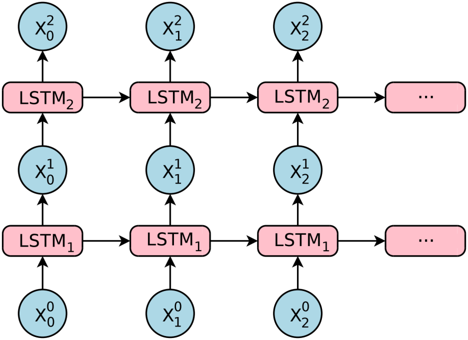


- Simply stacking layers on top of each other
- Harder to train, but found better than single-layer one[10]

[10] https://arxiv.org/pdf/1409.3215.pdf

Image credit: Yonghui Wu *et al.*

Deep RNNs

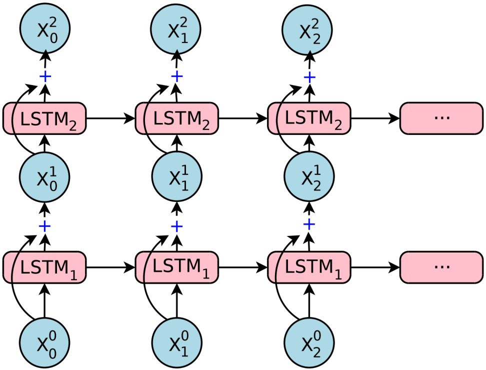


- Simply stacking layers on top of each other
- Harder to train, but found better than single-layer one[10]
- Gradients exploding or vanishing problem

[10] https://arxiv.org/pdf/1409.3215.pdf

Image credit: Yonghui Wu *et al.*

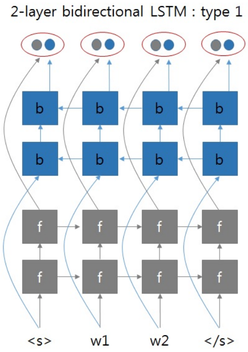
Deep RNNs with Residuals

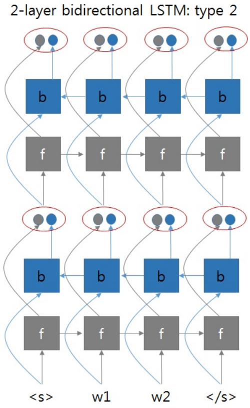


- Address gradient exploding or vanishing

Image credit: Yonghui Wu *et al.*

Deep, Bidirectional and ...





- Can combine deep with bidirectional
- Can vary with many options

Image credit: Sung Sue Hwang