

# Introduction to Deep Learning

## Lecture 8 RNNs and LSTMs

Maria Vakalopoulou & Stergios Christodoulidis



CentraleSupélec

**MICS Laboratory**

**CentraleSupélec**  
**Université Paris-Saclay**

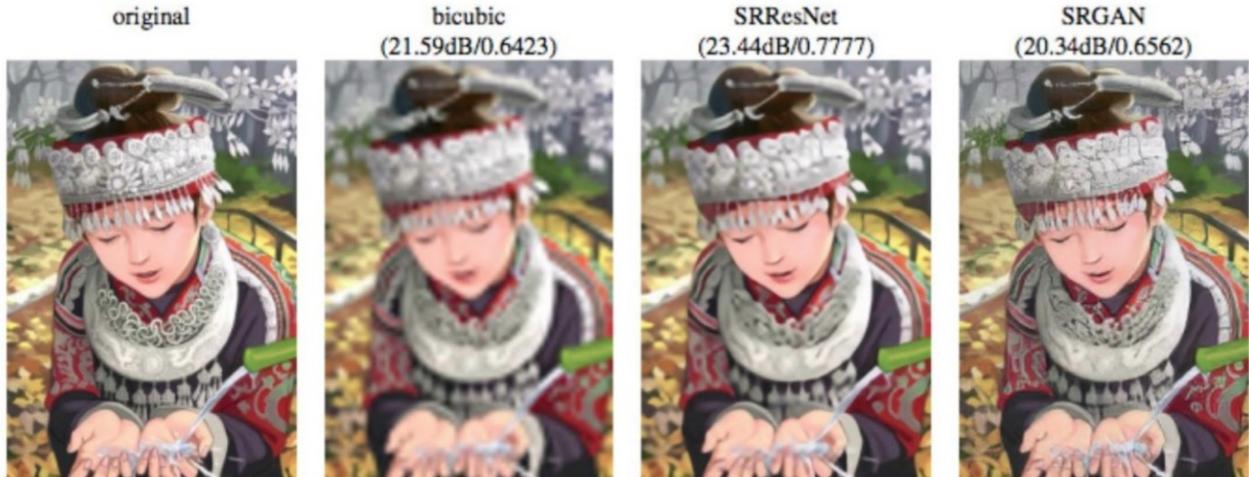
Wednesday, December 15, 2021

<https://stergioc.github.io/>

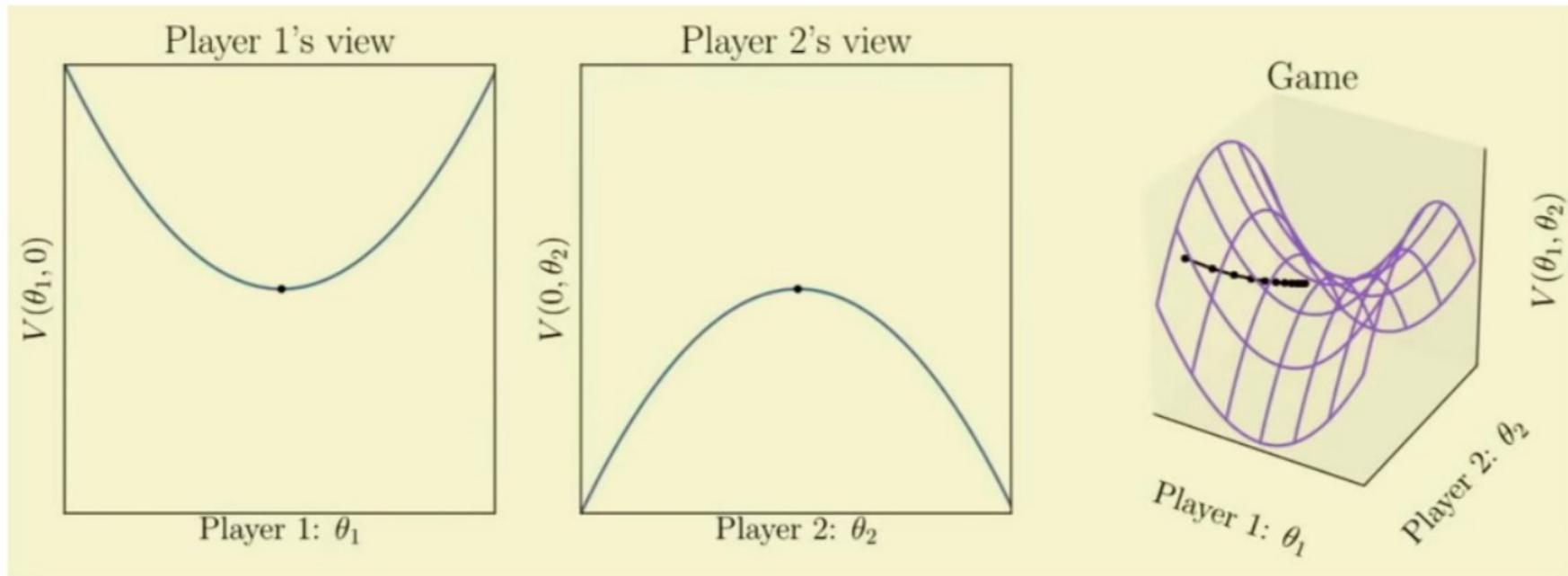


# Last Lecture

# Applications of Generative Models



# Adversarial Training (Min-Max Game)



Goodfellow, 2019

# Generative Adversarial Networks

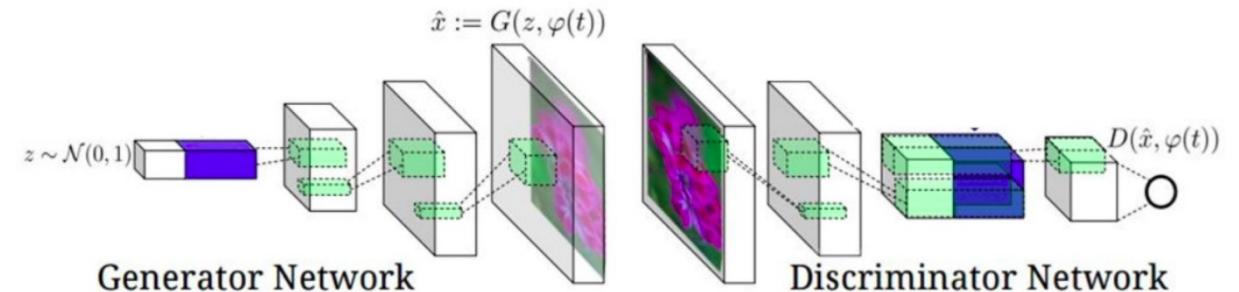
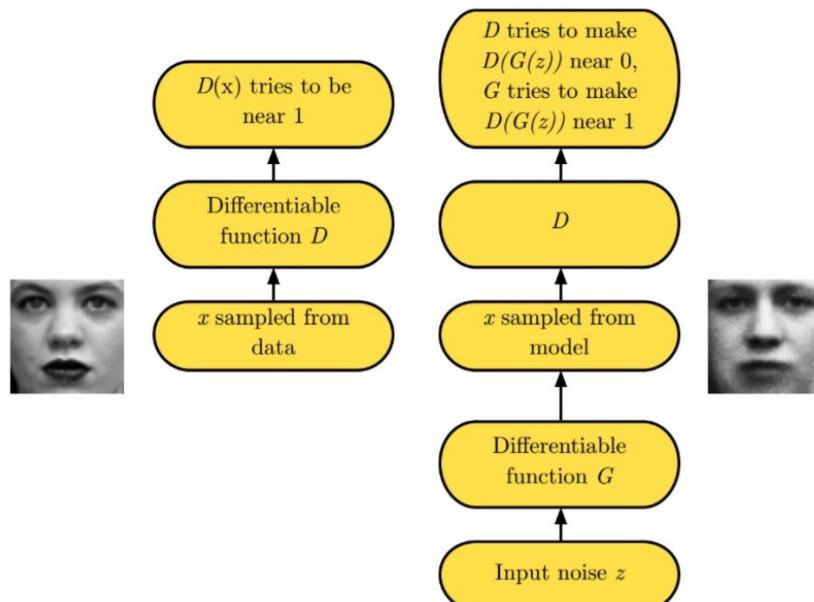
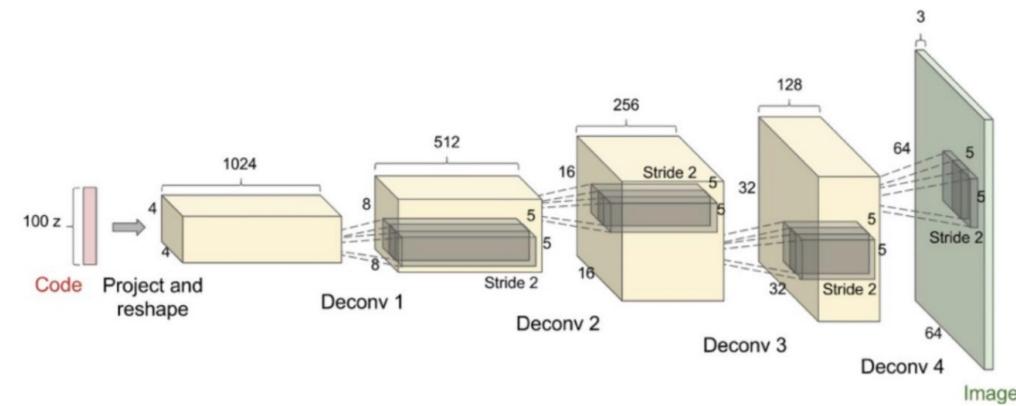


Figure 2. Our text-conditioned convolutional GAN architecture. Text encoding  $\varphi(t)$  is used by both generator and discriminator. It is projected to a lower-dimensions and depth concatenated with image feature maps for further stages of convolutional processing.



# Today's Lecture

# Today's Lecture

- Working with Sequential Data
- RNN models
- Backprop through time (BPTT)
- Vanishing/Exploding Gradients
- LSTMs
- Examples

# Motivation

- Up until now we have been looking at independent and identically distributed (i.i.d.) signals
- E.g. CNNs are specialized on processing grid of values (images)
- Imagine you want to predict what kind of activity is happening in a movie using a single frame.
- Or similarly, imagine you want to predict the topic of a sentence/paragraph using a single word/sentence.
  - E.g. “year”
  - E.g. “It was the previous year when I first realized it.”
- A large amount of the real world tasks rely on sequential data (e.g. Human attention)



# Sequential Data

- Videos
- Text
- Time Series
  - Stock prices
  - Biological measurements (heart rate, EEG, etc.)
  - Climate measurements (temperature, humidity, etc.)
- Speech/Music
- Human Behavior
- ...

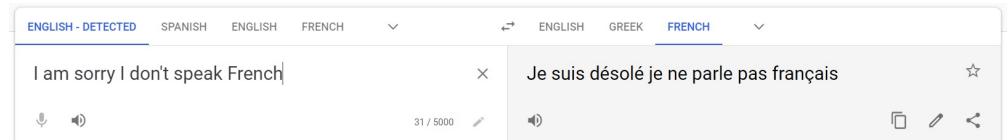
# How are sequences different?

- Sequences are of arbitrary or even infinite length.
- Data in sequences are not i.i.d.
  - Next word depends on previous words
- In order to answer questions that rely on sequential data we need some notion of memory.
  - **Data:** “In the year 2019, the SARS-CoV-2 virus emerged and quickly spread around the world. The World Health Organization (WHO) declared the outbreak a Public Health Emergency of International Concern on 30 January 2020, and a pandemic on 11 March 2020.”
  - **Question?**

→ **Recurrent Neural Networks (RNNs) are often used for handling sequential data**

# Applications of RNNs?

- Machine translation
- Video/Image captioning
- Question answering
- Video generation
- Speech synthesis
- Speech recognition
- ...



The screenshot shows a machine translation application with tabs for English, Spanish, English, and French. The English tab is selected. The input text is "I am sorry I don't speak French". The output text is "Je suis désolé je ne parle pas français". There are icons for microphone, speaker, and edit.



"man in black shirt is playing guitar."



"construction worker in orange safety vest is working on road."



"two young girls are playing with lego toy."

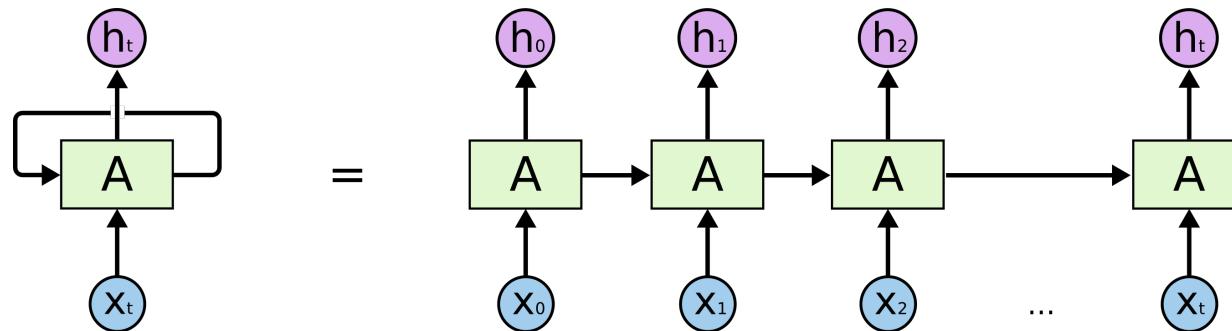
*Foreign Minister.* → FOREIGN MINISTER.



→ THE SOUND OF

# Recurrent Neural Networks (RNNs)

- They were first introduced in 1986.
- They are neural networks with loops in order to allow information to persist.
- These loops represent the influence of previous value on the same value at the current step.
- For a simpler representation we could unroll this RNN in time.



(<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>)

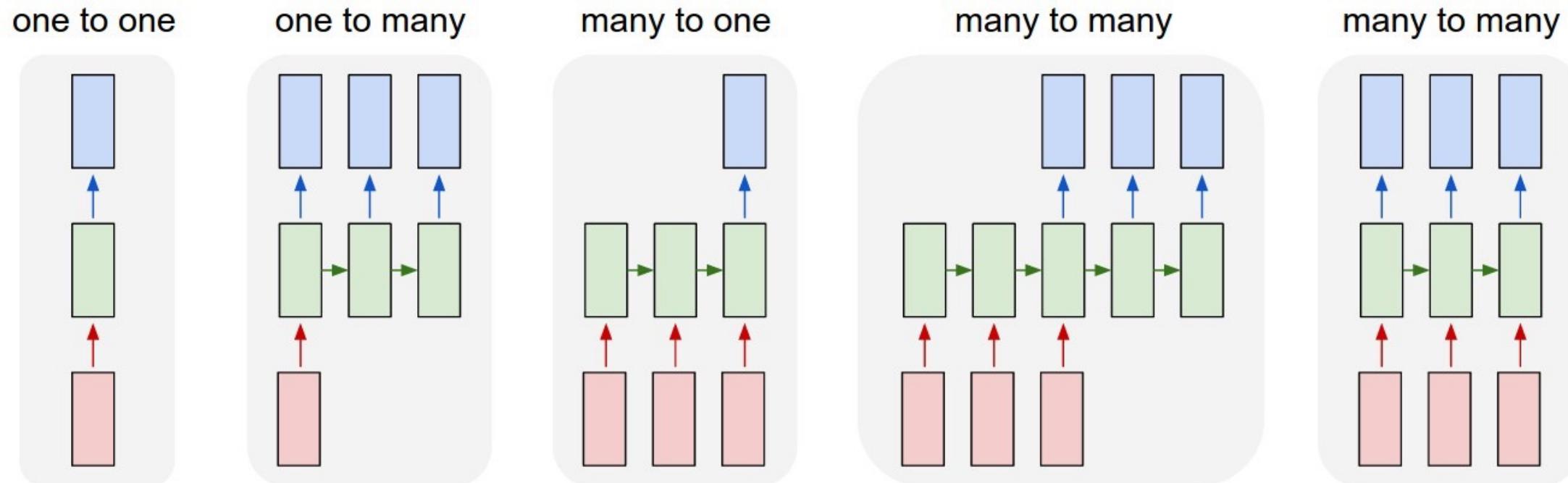
# Properties of Sequences

- We cannot, and ideally should not, have a constraint over the length of our sequence.
- Our model should have no problem with:
  - Varying
  - Unknown
  - Or even infinite Sequence lengths

→ **Parameter sharing makes it possible to extend and apply the model to examples of different forms and generalize across them.**

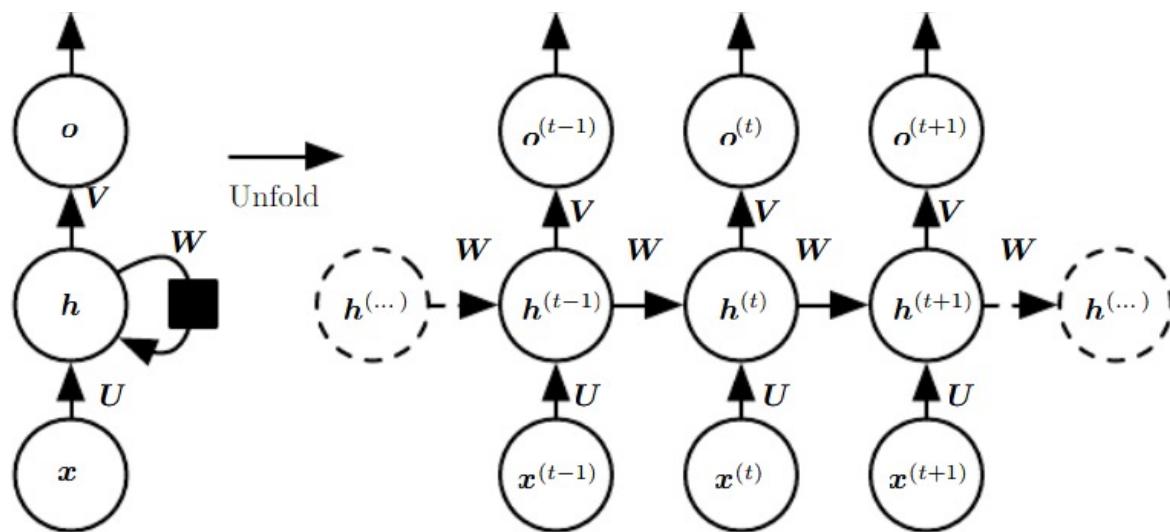
- It can be particularly important when a specific piece of information can occur multiple times or at different positions
  - “Answer to the Ultimate Question of Life, The Universe, and Everything”
  - “My name is Bond, James Bond”

# Input-Output Schemes of RNNs



Andrey Karpathy, "The Unreasonable Effectiveness of Recurrent Neural Networks", 2015

# Example Recurrent Neural Network

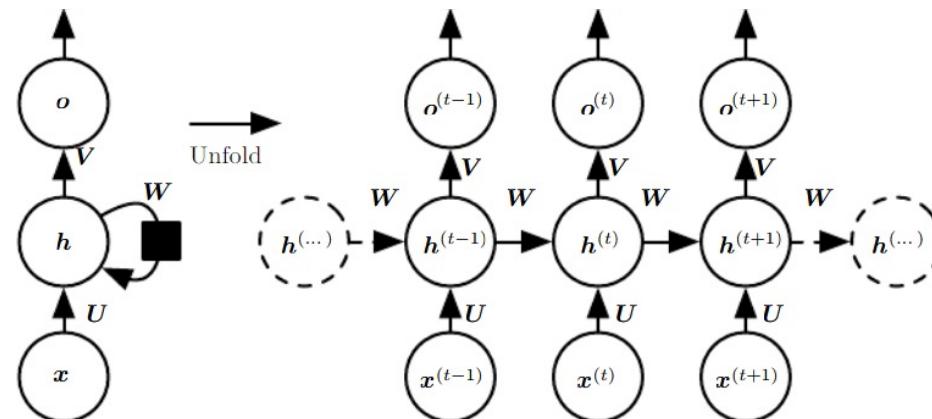


$$\mathbf{h}^{(t)} = \tanh(\mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)})$$

$$\mathbf{y}^{(t)} = \text{softmax}(\mathbf{V}\mathbf{h}^{(t)})$$

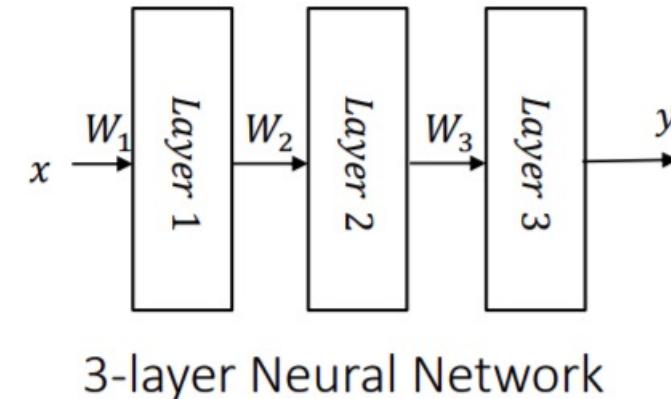
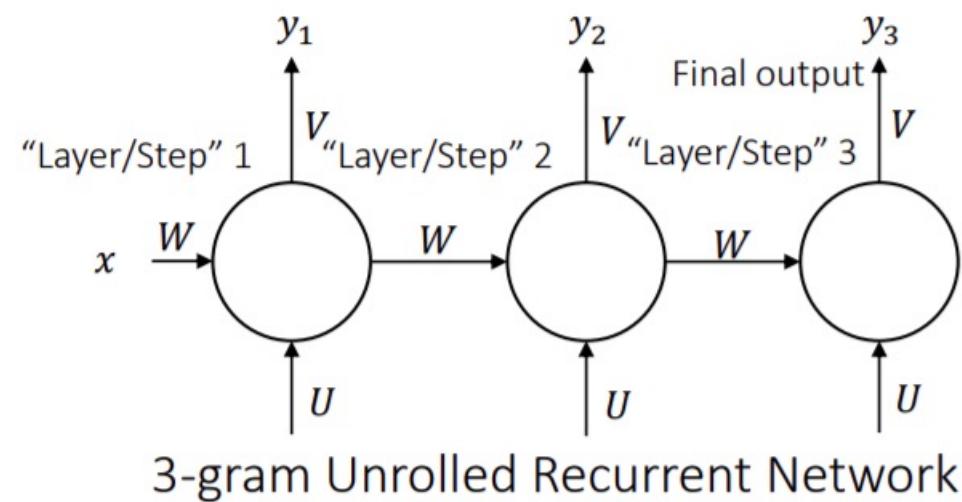
# Example Recurrent Neural Network

- In the simplest case with fully connected layers  $\theta = \{U, V, W\}$
- We map our inputs to the latent representation through  $U$
- This map our current latent representation to the new one through  $W$
- The output is computed through the set of parameters  $V$
- At each timestep  $t$  all previous information  $1, \dots, t$  is used through the recurrent connection.



# RNN vs MLP

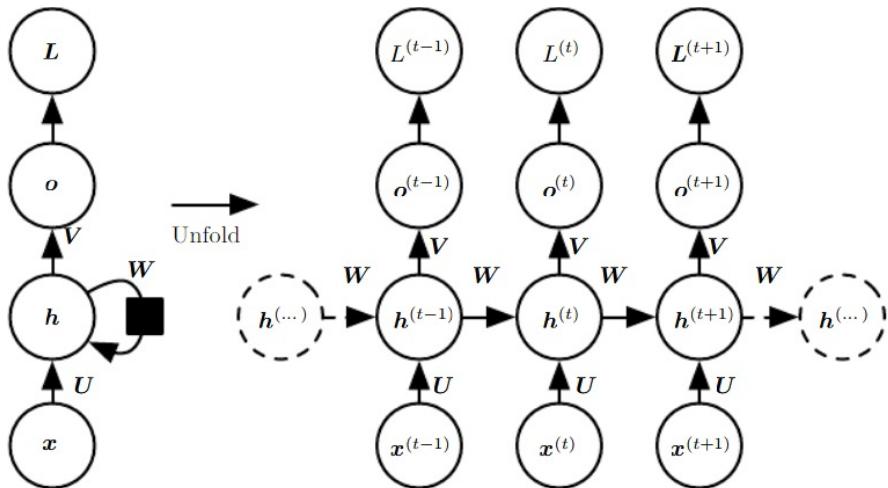
- Instead of layers → Steps
- Output in every step → MLP outputs in every layer possible
- Main difference: Instead of layer specific parameters → Layer shared parameters



# Backpropagation Through Time (BPTT)

- Using the generalized backpropagation algorithm, and by unrolling the RNN we can obtain the back propagation through time algorithm.
- Basically, we can use the chain rule again on the unfolded computational graph.
- We just need to take into consideration that gradients survive through time.
- We need to calculate the gradient of the Loss function with respect to our parameters:  $\frac{\partial \mathcal{L}}{\partial U}, \frac{\partial \mathcal{L}}{\partial V}, \frac{\partial \mathcal{L}}{\partial W}$
- $U, V, W$  are the same for all  $t$
- The gradient flows from multiple paths in contrast with MLPs

# Backpropagation Through Time (BPTT)



$$\mathbf{h}^{(t)} = \tanh(\mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)})$$

$$\mathbf{o}^{(t)} = \text{softmax}(\mathbf{V}\mathbf{h}^{(t)})$$

$$\mathcal{L} = \sum_t \mathcal{L}^{(t)} = \sum_t \mathcal{L}^{(t)} (\mathbf{o}^{(t)}, \mathbf{y}^{(t)})$$

$$\frac{\partial \mathcal{L}}{\partial V} = \frac{\partial \mathcal{L}}{\partial \mathcal{L}^{(t)}} \cdot \frac{\partial \mathcal{L}^{(t)}}{\partial \mathbf{o}^{(t)}} \cdot \frac{\partial \mathbf{o}^{(t)}}{\partial V}$$

$$\frac{\partial \mathcal{L}}{\partial W} = \frac{\partial \mathcal{L}}{\partial \mathcal{L}^{(t)}} \cdot \frac{\partial \mathcal{L}^{(t)}}{\partial \mathbf{o}^{(t)}} \cdot \frac{\partial \mathbf{o}^{(t)}}{\partial \mathbf{h}^{(t)}} \cdot \frac{\partial \mathbf{h}^{(t)}}{\partial W}$$

$$\frac{\partial \mathcal{L}}{\partial U} = \frac{\partial \mathcal{L}}{\partial \mathcal{L}^{(t)}} \cdot \frac{\partial \mathcal{L}^{(t)}}{\partial \mathbf{o}^{(t)}} \cdot \frac{\partial \mathbf{o}^{(t)}}{\partial \mathbf{h}^{(t)}} \cdot \frac{\partial \mathbf{h}^{(t)}}{\partial U}$$

# BPTT: Chain Rule for $\partial \mathcal{L} / \partial W$

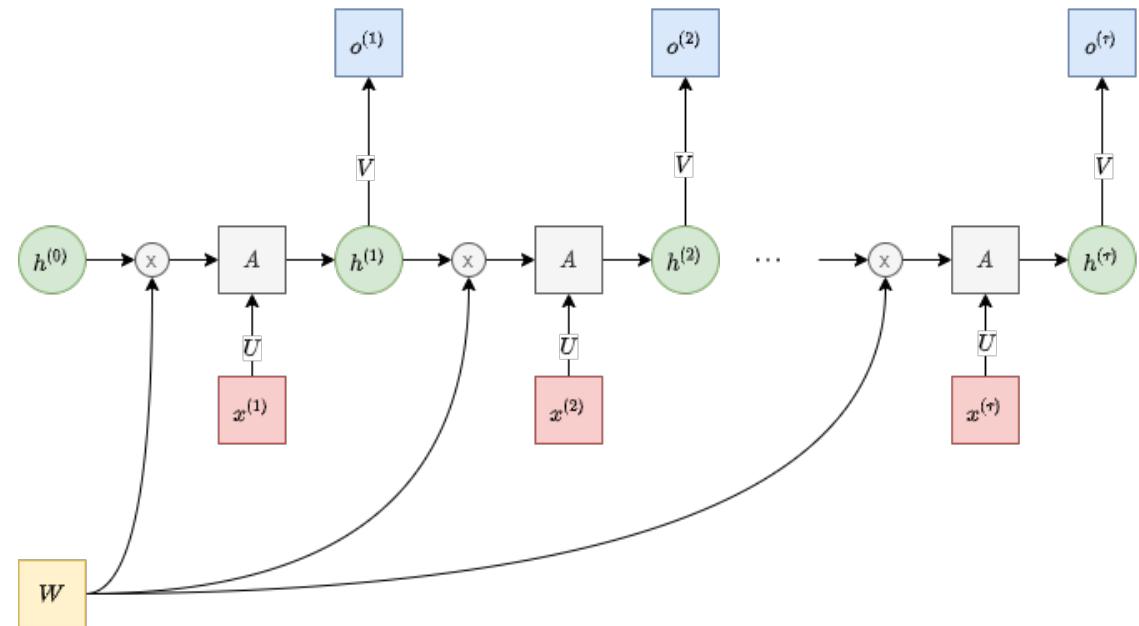
- All gradient path flows from  $h$  to  $W$ 
  - Via  $h^{(\tau)}$
  - Via  $h^{(\tau-1)}$
  - ...

- Putting everything together:

$$\frac{\partial \mathcal{L}(t)}{\partial W} = \sum_{i=0}^t \frac{\partial \mathcal{L}(t)}{\partial o^{(t)}} \frac{\partial o^{(t)}}{\partial h^{(t)}} \frac{\partial h^{(t)}}{\partial h^{(i)}} \frac{\partial h^{(i)}}{\partial W}$$

- If there are multiple losses over time steps:

$$\frac{\partial \mathcal{L}}{\partial W} = \sum_t \frac{\partial \mathcal{L}(t)}{\partial W}$$



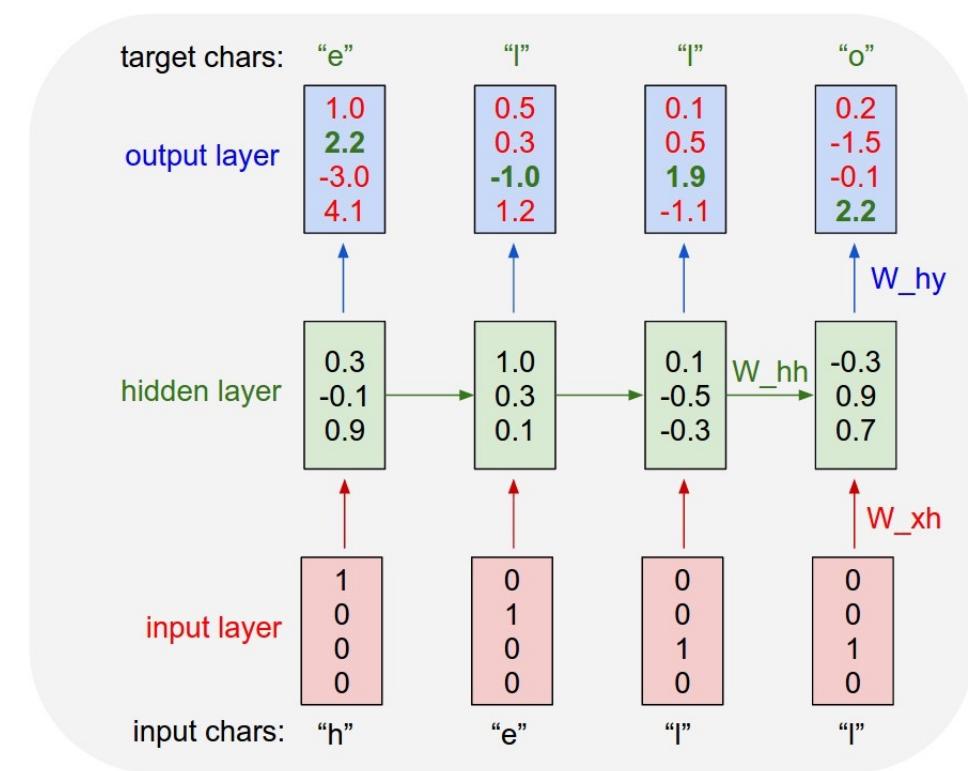
# min-char-rnn.py

```
min-char-rnn.py
Raw
56     dikh += np.dot(dhraw, xs[t].T)
57     dihh += np.dot(dhraw, hs[t-1].T)
58     dhnext = np.dot(dihh.T, dhraw)
59     for dparam in [dikh, dihh, dhy, dbh, dby]:
60         np.clip(dparam, -5, 5, out=dparam) # clip to mitigate exploding gradients
61     return loss, dikh, dihh, dhy, dbh, dby, hs[len(inputs)-1]
62
63 def sample(h, seed_ix, n):
64     """
65     sample a sequence of integers from the model
66     h is memory state, seed_ix is seed letter for first time step
67     """
68     x = np.zeros((vocab_size, 1))
69     x[seed_ix] = 1
70     ixes = []
71     for t in xrange(n):
72         h = np.tanh(np.dot(wkh, x) + np.dot(bkh, h) + bh)
73         y = np.dot(why, h) + by
74         p = np.exp(y) / np.sum(np.exp(y))
75         ix = np.random.choice(range(vocab_size), p=p.ravel())
76         x = np.zeros((vocab_size, 1))
77         x[ix] = 1
78         ixes.append(ix)
79     return ixes
80
81 n, p = 0, 0
82 mixh, mihh, mihy = np.zeros_like(ixh), np.zeros_like(ihh), np.zeros_like(ihy)
83 mbh, mby = np.zeros_like(bh), np.zeros_like(by) # memory variables for Adagrad
84 smooth_loss = -np.log(1.0/vocab_size)*seq_length # loss at iteration 0
85 while True:
86     # prepare inputs (we're sweeping from left to right in steps seq_length long)
87     if p+seq_length1 > len(data) or n == 0:
88         hprev = np.zeros((hidden_size,1)) # reset RNN memory
89         p = 0 # go from start of data
90     inputs = [char_to_ix[ch] for ch in data[p:p+seq_length]]
91     targets = [char_to_ix[ch] for ch in data[p+1:p+seq_length+1]]
92
93     # sample from the model now and then
94     if n % 100 == 0:
95         sample_ix = sample(hprev, inputs[0], 200)
96         txt = ''.join(ix_to_char[i] for ix in sample_ix)
97         print '----\n% s\n----' % (txt, )
98
99     # forward seq_length characters through the net and fetch gradient
100    loss, dikh, dihh, dhy, dbh, dby, hprev = lossfun(inputs, targets, hprev)
101    smooth_loss = smooth_loss * 0.999 + loss * 0.001
102    if n % 100 == 0: print 'iter %d, loss: %f' % (n, smooth_loss) # print progress
103
104    # perform parameter update with Adagrad
105    for param, dparam, mem in zip([ixh, ihh, ihy, bh, by],
106                                [dikh, dihh, dhy, dbh, dby],
107                                [mixh, mihh, mihy, mbh, mby]):
108        mem += dparam * dparam
109        param -= learning_rate * dparam / np.sqrt(mem + 1e-8) # adagrad update
110
111    p += seq_length # move data pointer
112    n += 1 # iteration counter
```

<https://gist.github.com/karpathy/d4dee566867f8291f086>

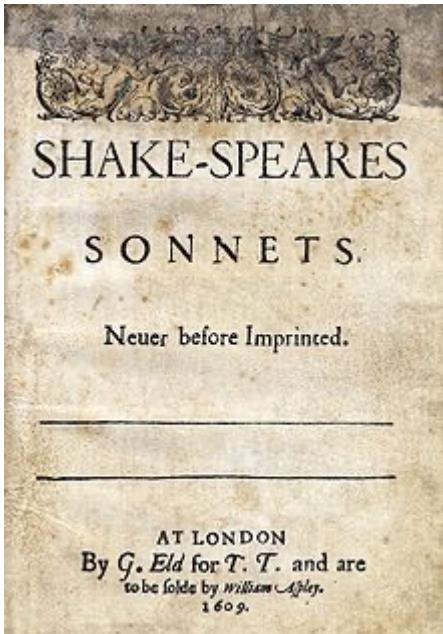
# Example: Character Level Language Models

- Example of a simple RNN used in a character level
- Consider a huge text chunk
- If we encode each character to an one-hot vector.
- We can train an RNN to predict the probability distribution of the next character
- As such, we can then generate a text a character at a time.
- At inference time, we sample from the output distribution in order to select the next character



Andrej Karpathy, "The Unreasonable Effectiveness of Recurrent Neural Networks", 2015

# Shakespeare Database



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

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

we counter. He stutn co des. His stanted out one ofler that concossions and was  
to gearang reay Jotrets and with fre colt oft paitt thin wall. Which das stimn

"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.

Andrej Karpathy, "The Unreasonable Effectiveness of Recurrent Neural Networks", 2015

# Shakespeare Database

PANDARUS:

Alas, I think he shall be come approached and the day  
When little strain 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.

Andrej Karpathy, "The Unreasonable Effectiveness of Recurrent Neural Networks", 2015

# Linux Source Code Database

```
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
    int error;
    if (fd == MARN_EPT) {
        /*
         * The kernel blank will coeld it to userspace.
         */
        if (ss->segment < mem_total)
            unblock_graph_and_set_blocked();
    } else
        ret = 1;
    goto bail;
}
segaddr = in_SB(in.addr);
selector = seg / 16;
setup_works = true;
for (i = 0; i < blocks; i++) {
    seq = buf[i++];
    bpf = bd->bd.next + i * search;
    if (fd) {
        current = blocked;
    }
}
rw->name = "Getjbbregs";
bprm_self_clear(&iv->version);
regs->new = blocks[(BPF_STATS << info->historidac)] | PFMR_CLOBATHINC_SECONDS << 12;
return segtable;
}
```

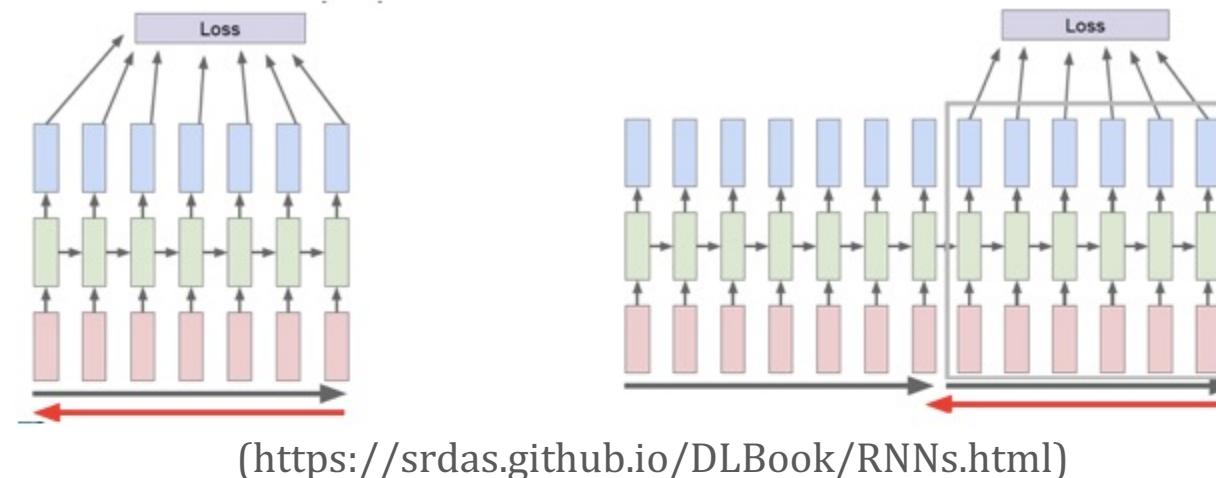
Andrej Karpathy, “The Unreasonable Effectiveness of Recurrent Neural Networks”, 2015

# Standard RNNs are Difficult to Train in Practice

- Unrolling forever is not practical or even feasible
- Gradients propagated over many stages tend to either vanish (most of the time) or explode (rarely)
- Long-term memory difficult to persist.
  - Related with the vanishing gradients problem.

# Truncating BPTT

- Truncating to some  $\tau$  ( $\sim 100$ ) is a usual strategy.
- Analogous to stochastic gradient descent.
- You can carry the hidden state variables of the previous batch to the current
- More focus on short-term memory
  - Not necessarily undesirable, as long term dependency might be irrelevant
  - Biases towards simpler models with shorter-term dependencies



# Vanishing and Exploding Gradients

- Gradients vanish or explode even easier because of shared parameters

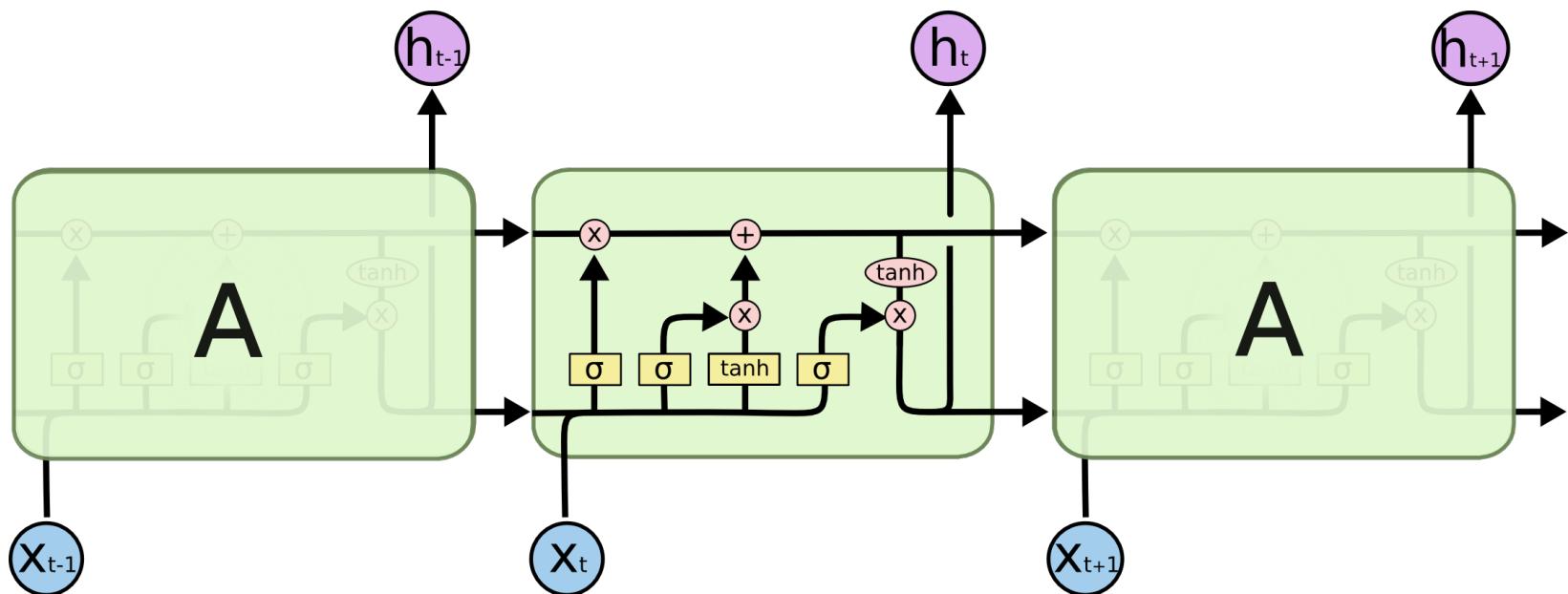
$$\frac{\partial \mathcal{L}^{(t)}}{\partial W} = \sum_{i=0}^t \frac{\partial \mathcal{L}^{(t)}}{\partial o^{(t)}} \frac{\partial o^{(t)}}{\partial h^{(t)}} \frac{\partial h^{(t)}}{\partial h^{(i)}} \frac{\partial h^{(i)}}{\partial W} = \sum_{i=0}^t \frac{\partial \mathcal{L}^{(t)}}{\partial o^{(t)}} \frac{\partial o^{(t)}}{\partial h^{(t)}} \left( \prod_{j=i+1}^t \frac{\partial h^{(j)}}{\partial h^{(j-1)}} \right) \frac{\partial h^{(i)}}{\partial W}$$

- If  $\frac{\partial h^{(j)}}{\partial h^{(j-1)}} > 1 \Rightarrow \frac{\partial \mathcal{L}^{(t)}}{\partial W} \gg 1 \rightarrow$  Exploding Gradient → Gradient Clipping
- If  $\frac{\partial h^{(j)}}{\partial h^{(j-1)}} < 1 \Rightarrow \frac{\partial \mathcal{L}^{(t)}}{\partial W} \ll 1 \rightarrow$  Vanishing Gradient → Architecture Change

# Long short-term memory (LSTMs)

- LSTMs are an RNN architecture.
- They are capable of learning long-term dependencies
- Tackling the vanishing gradient problem
- The LSTM has been found extremely successful in many applications

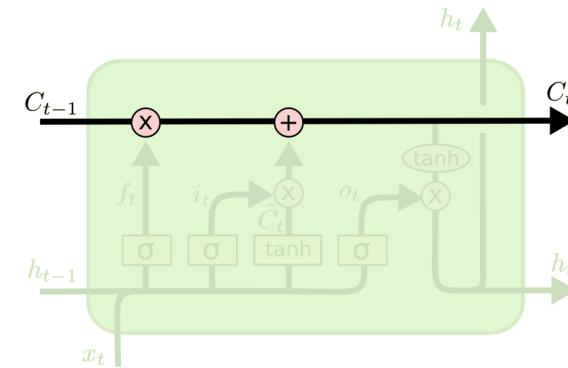
# LSTMs



(<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>)

# LSTM: Cell State

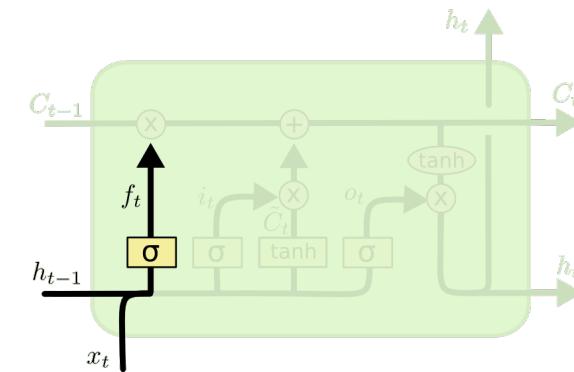
- The core idea behind LSTMs lies on the cell state (the recurrent part).
- It runs straight down the entire module.
- Minor interactions with the other components.
- It is possible that information flow unchanged.
- The LSTM has the ability to add/remove information from this cell state in a regulated manner.



(<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>)

# LSTM: Forget Gate

- The first step is to decide what information we are going to throw away.
- It looks at the previous output and the current input.
- The sigmoid brings the values to [0,1] ranges
- When multiplied with the cell state the relevant information will be kept and the rest will be discarded
- This is called the forget gate.

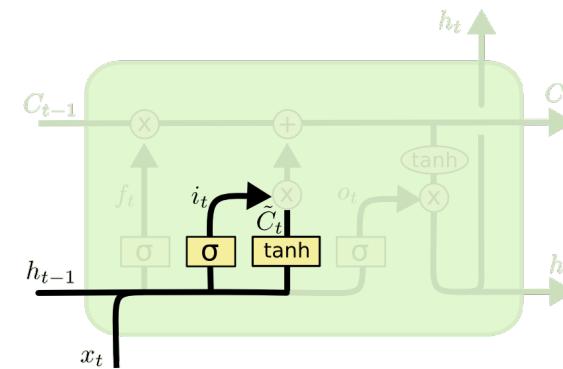


$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

(<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>)

# LSTM: Input Gate

- Before generating the new cell state:
  - Then we identify new candidate values that could be added in the cell state from the new input
  - We scale them with another sigmoid
- We are again looking at the previous output and the current input.



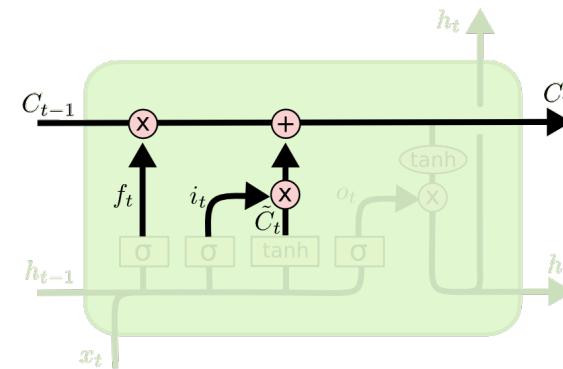
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

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

(<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>)

# LSTM: New Cell State

- Now we can apply the forget and write operation in the cell state
- We achieve this with a simple multiplication and addition.
  - Multiplying with the forget signal
  - Adding the new input signal

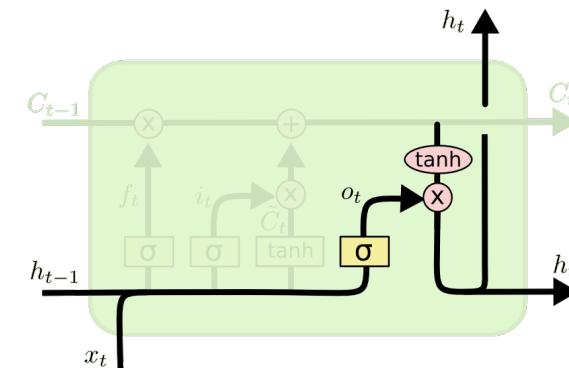


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

(<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>)

# LSTM: Output

- Lastly, we can use the new cell state to calculate the output
- We can push the values to the range [-1,1] using the tanh
- We select the parts that are relevant for our output using a sigmoid

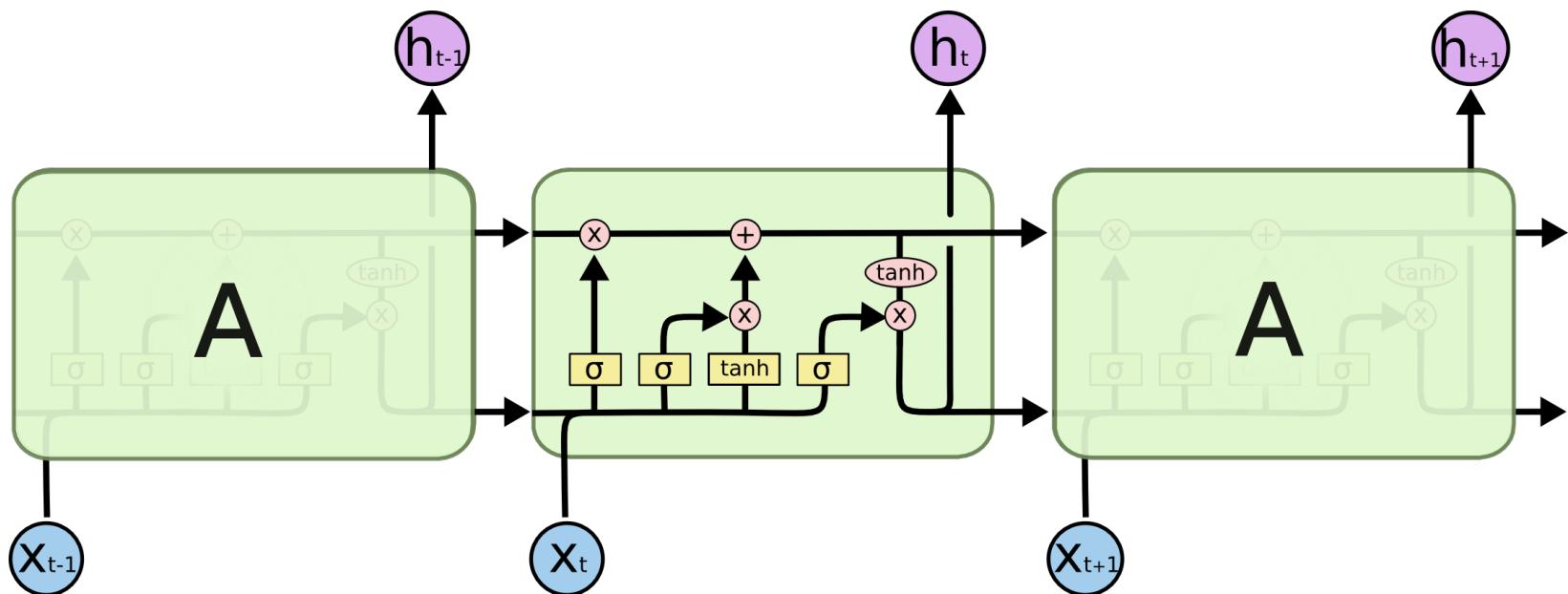


$$o_t = \sigma(W_o [h_{t-1}, x_t] + b_o)$$

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

(<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>)

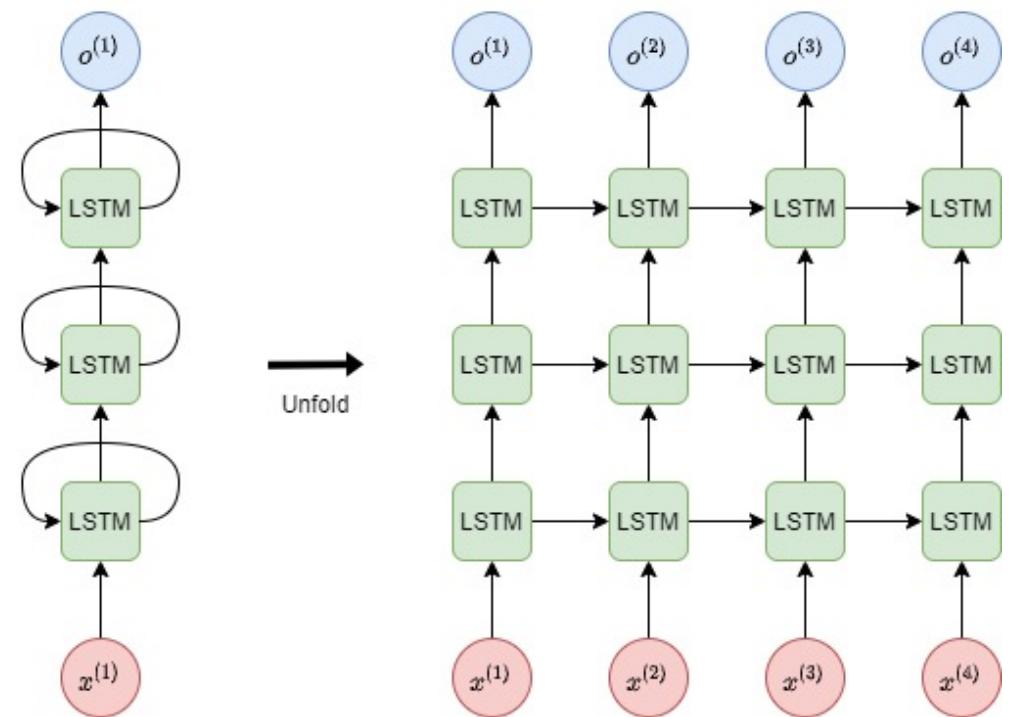
# LSTMs



(<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>)

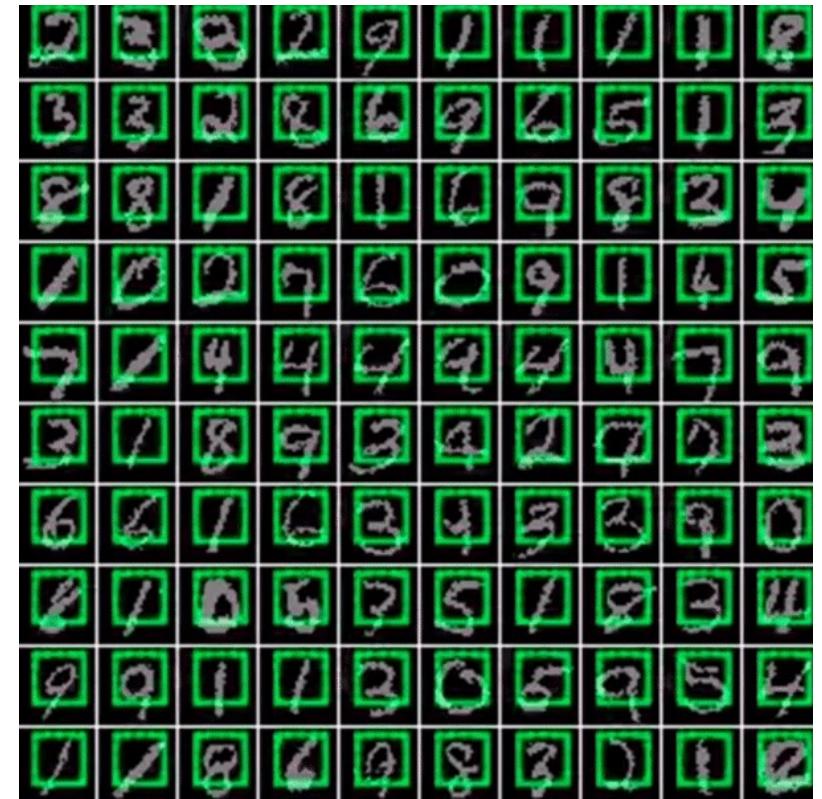
# Multi-Layer LSTMs

- RNNs can be stacked in order to form multi-layer architectures.
- In this case we have depth and time.
- BPTT still stands and these can also be optimized
- For many problems you might see 2-4 RNN layers stacked on top of each other



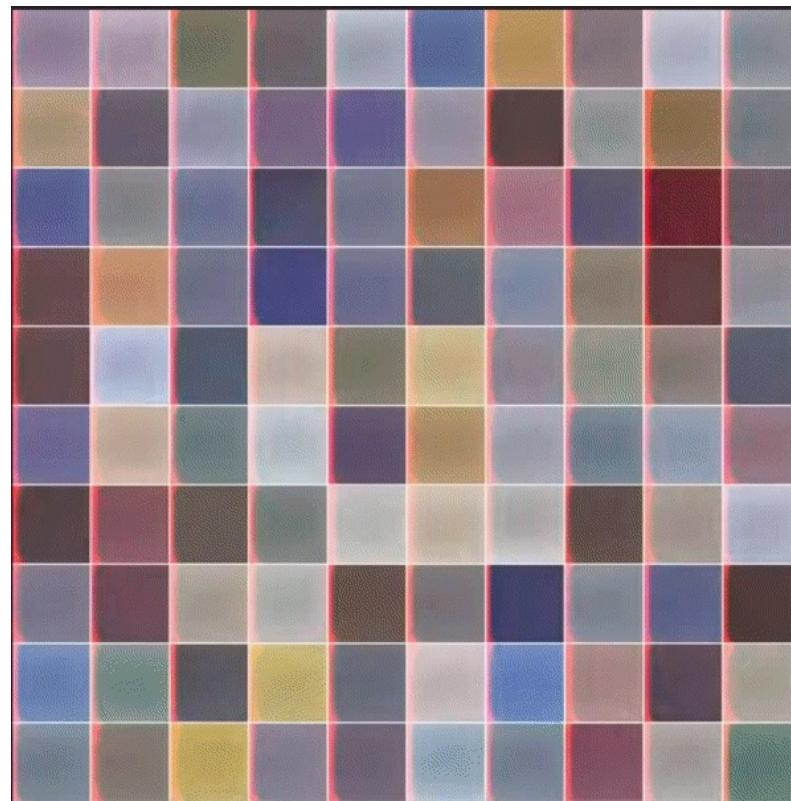
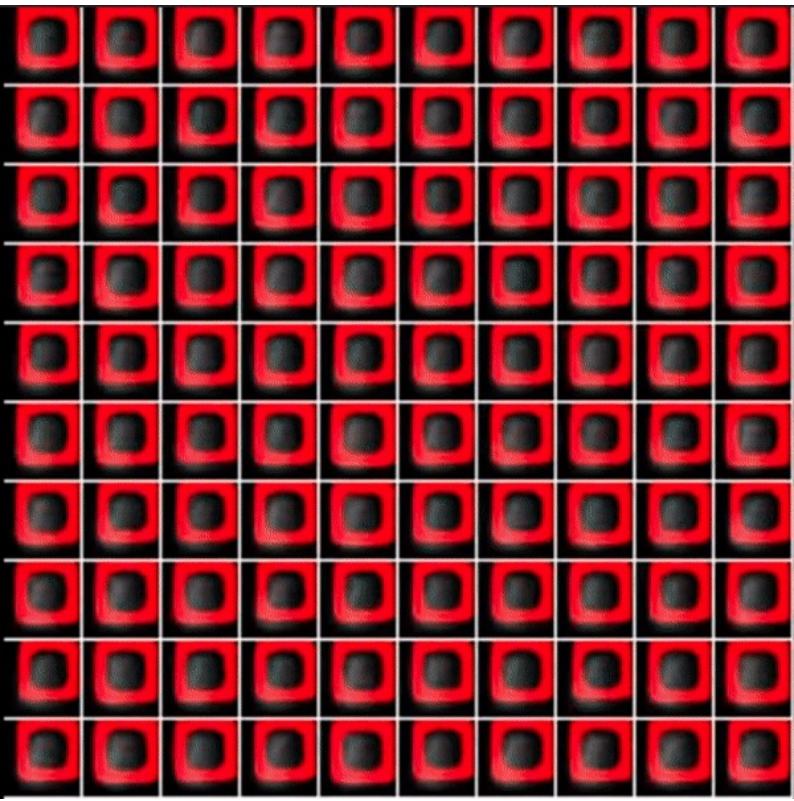
# LSTM for Classification

- An LSTM RNN is trained in order to selectively get glimpses of the images
- These glimpses are received in a predefined size (12x12)
- The glimpse sampling is defined in a differential way.
- After a pre-defined number of glimpses the RNN is using the final cell state to perform the classification



Gregor, Karol, et al. "Draw: A recurrent neural network for image generation", ICLR 2015

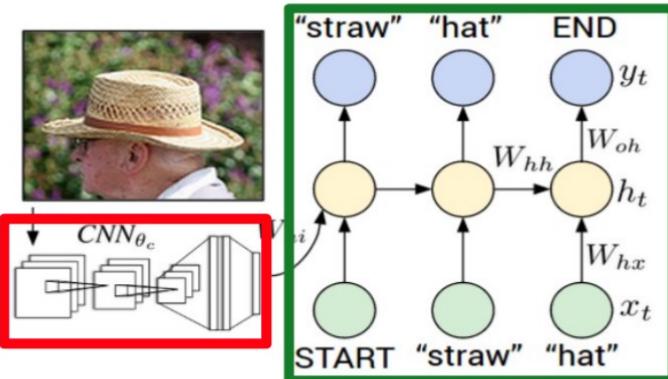
# LSTM for Generation



Gregor, Karol, et al. "Draw: A recurrent neural network for image generation", ICLR 2015

# Image Captioning

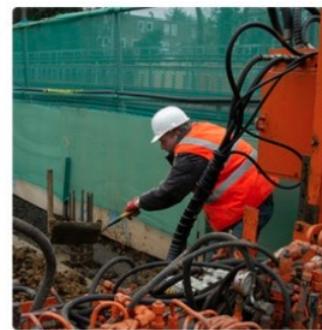
## Recurrent Neural Network



## Convolutional Neural Network



"man in black shirt is playing guitar."



"construction worker in orange safety vest is working on road."



"two young girls are playing with lego toy."



"boy is doing backflip on wakeboard."



"girl in pink dress is jumping in air."



"black and white dog jumps over bar."



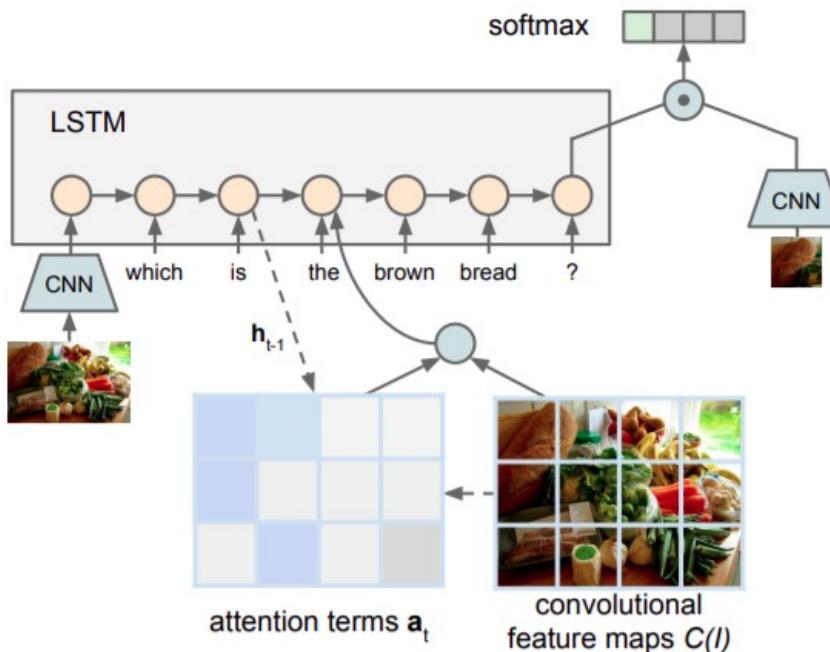
"young girl in pink shirt is swinging on swing."



"man in blue wetsuit is surfing on wave."

Andrej Karpathy, "Automated Image Captioning with ConvNets and Recurrent Nets", 2015

# Visual Question Answering



**Q:** Which is the brown bread?

- A1:**    **A2:**    **A3:**    **A4:**



**Q:** What endangered animal is featured on the truck?

- A:** A bald eagle.  
**A:** A sparrow.  
**A:** A humming bird.  
**A:** A raven.

**Q:** Where will the driver go if turning right?

- A:** Onto 24 ½ Rd.  
**A:** Onto 25 ¼ Rd.  
**A:** Onto 23 ¾ Rd.  
**A:** Onto Main Street.

**Q:** When was the picture taken?

- A:** During a wedding.  
**A:** During a bar mitzvah.  
**A:** During a funeral.  
**A:** During a Sunday church service.



**Q:** Which pillow is farther from the window?

**Q:** Which step leads to the tub?

**Q:** Which is the small computer in the corner?

Yuke Zhu, et al., "Visual7W: Grounded Question Answering in Images", 2016

# Take Home Messages

- Standard RNNs have been proposed in order to handle sequential data.
- The core idea is the use of the recurrent connection to allow memory to persist.
- They can be trained using the backpropagation through time.
  - Chain rule still holds
- Due to the properties of the data and their structure they are difficult to train.
- LSTMs and variants have been introduced in order to solve some of these difficulties in training.
  - A lot of variants since then have revolutionized the field.
- Further Reading:
  - Deep Learning Book [Chapter 10](#), Ian Goodfellow, Yoshua Bengio, Aaron Courville
  - All the links and references that are provided in these slides