CS 1674/2074: Sequential Data: Language and Vision; Video and Motion

PhD. Nils Murrugarra-Llerena
nem177@pitt.edu

Plan for this lecture

- Language and vision
  - Application: Image and video captioning
  - Tool: Recurrent neural networks
  - Tool: Transformers
  - Application: Visual question answering
- Motion and video
  - Video classification
  - Measuring motion
  - Tracking objects

Motivation: Descriptive Text for Images

"It was an arresting face, pointed of chin, square of jaw. Her eyes were pale green without a touch of hazel, starred with bristly black lashes and slightly tilted at the ends. Above them, her thick black brows slanted upward, cutting a startling oblique line in her magnolia-white skin–that skin so prized by Southern women and so carefully guarded with bonnets, veils and mittens against hot Georgia suns"

Scarlett O'Hara described in Gone with the Wind

Some pre-RNN Good Results

This is a picture of one sky, one road and one sheep. The gray sky is over the gray road. The gray sheep is by the gray road.

This is a picture of two dogs. The first dog is near the second furry dog.

Here we see one road, one sky and one bicycle. The road is near the blue sky, and near the colorful bicycle. The colorful bicycle is within the blue sky.

Kulkarni et al., CVPR 2011

Some pre-RNN Good Results

Missed detections:

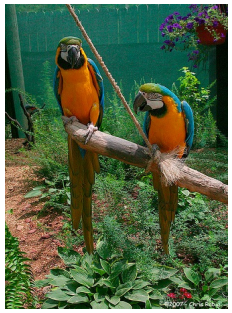
Here we see one potted plant.

This is a picture of one dog.

Kulkarni et al., CVPR 2011

False detections:

There are one road and one cat. The furry road is in the furry cat.



This is a picture of one tree, one road and one person. The rusty tree is under the red road. The colorful person is near the rusty tree, and under the red road.

Incorrect attributes:

This is a photograph of two sheeps and one grass. The first black sheep is by the green grass, and by the second black sheep. The second black sheep is by the green

This is a photograph of two horses and one grass. The first feathered horse is within the green grass, and by the second feathered horse. The second feathered horse is within the green grass.

Results with Recurrent Neural Networks

"man in black shirt is playing guitar."

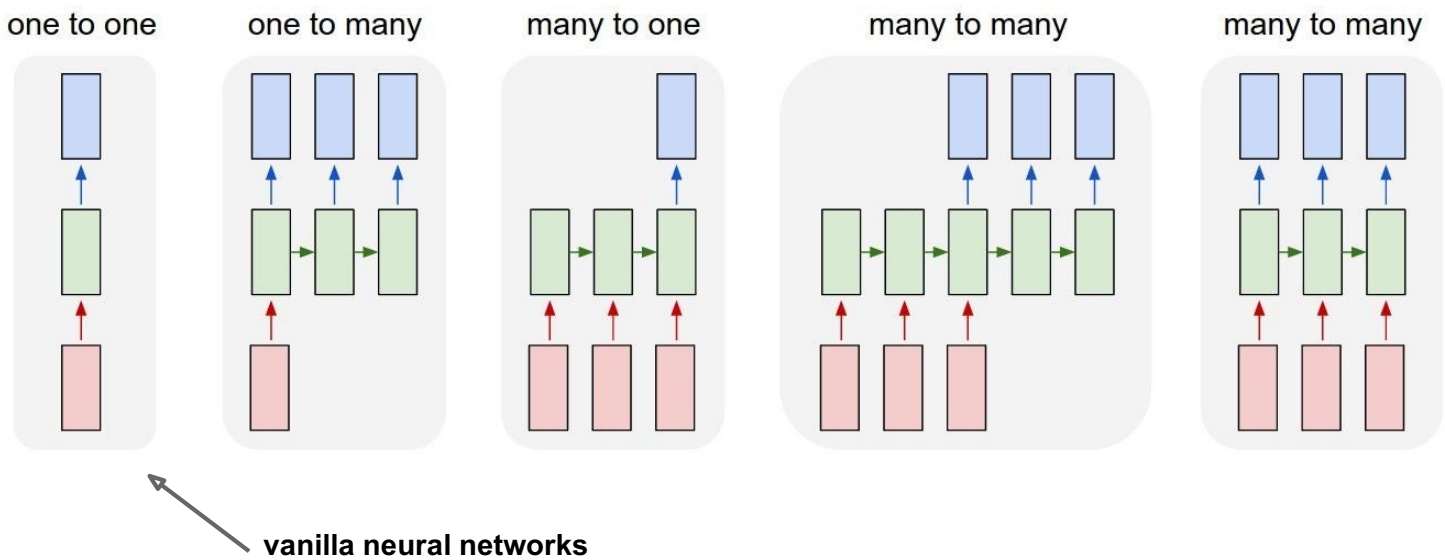
"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

"boy is doing backflip on wakeboard."

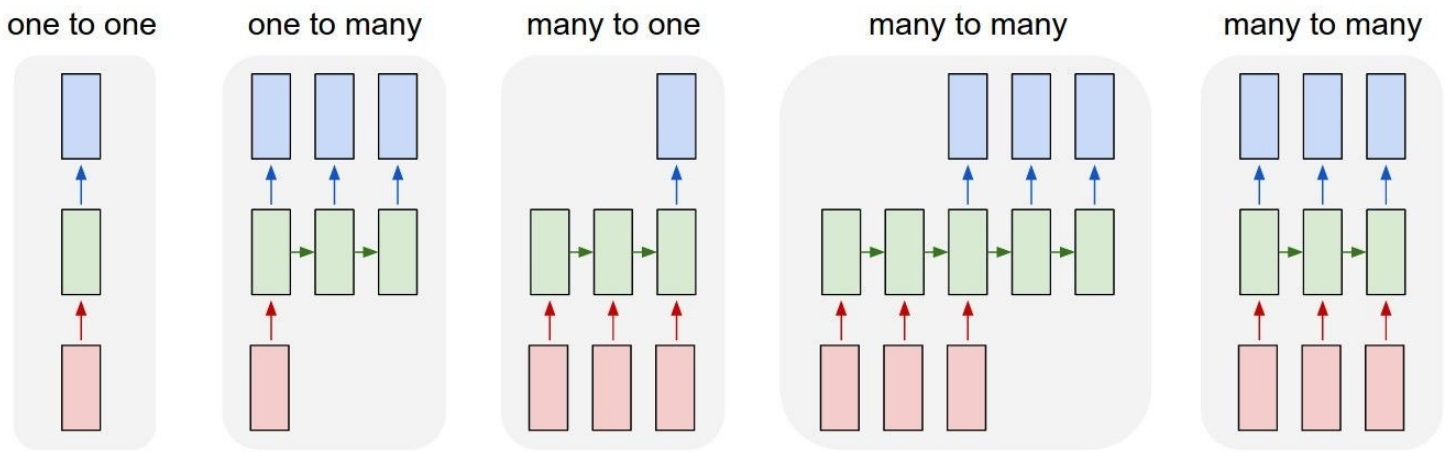
Karpathy and Fei-Fei, CVPR 2015

Recurrent Networks offer a lot of flexibility:



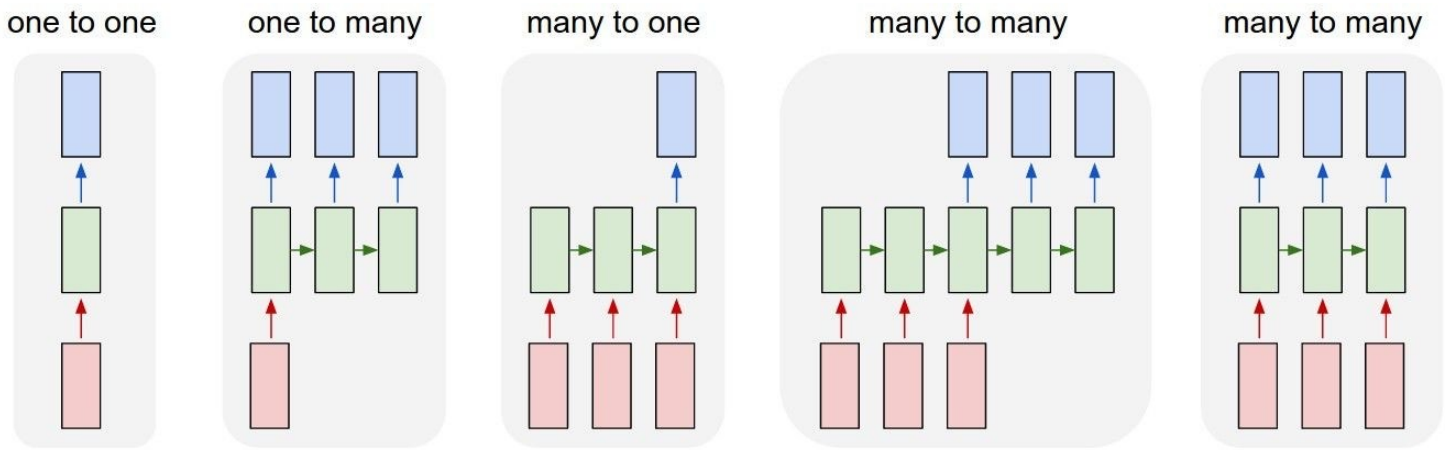
Andrej Karpathy

Recurrent Networks offer a lot of flexibility:



e.g. **image captioning**
image -> sequence of words

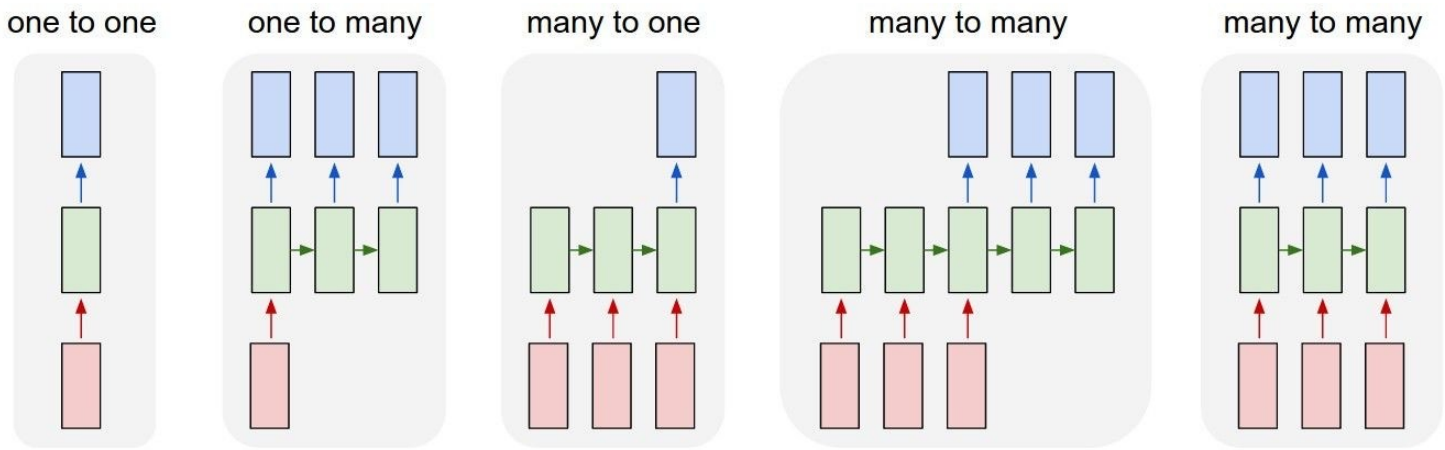
Recurrent Networks offer a lot of flexibility:



e.g. **sentiment classification**
sequence of words -> sentiment

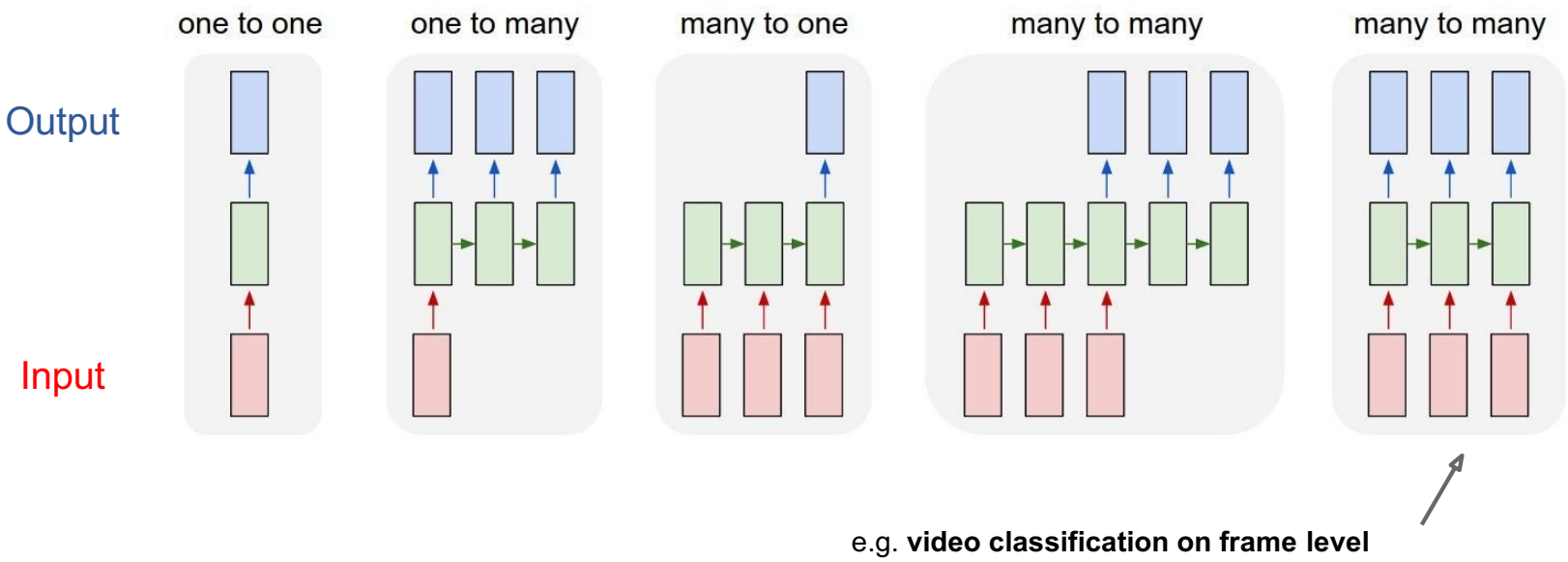
Andrej Karpathy

Recurrent Networks offer a lot of flexibility:



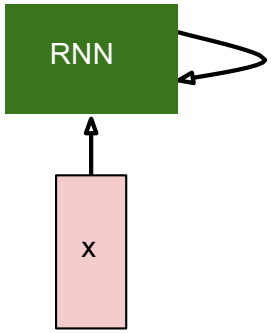
e.g. **machine translation**
seq of words -> seq of words

Recurrent Networks offer a lot of flexibility:

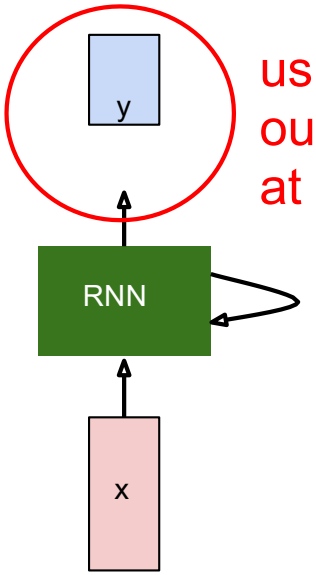


e.g. **video classification on frame level**

Recurrent Neural Network



Recurrent Neural Network



usually want to
output a prediction
at some time steps

Adapted from Andrej Karpathy

Recurrent Neural Network

We can process a sequence of vectors **x** by applying a recurrence formula at every time step:

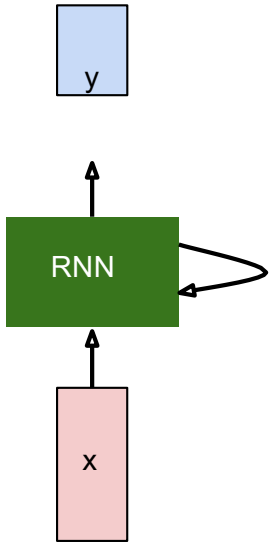
$$h_t = f_W(h_{t-1}, x_t)$$

new state

some function with parameters W

old state

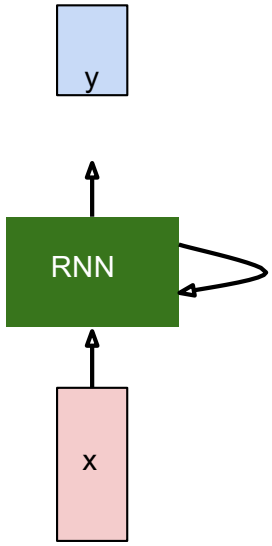
input vector at some time step



Recurrent Neural Network

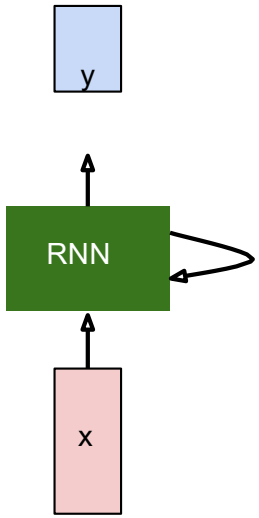
We can process a sequence of vectors **x** by applying a recurrence formula at every time step:

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



Notice: the same function and the same set of parameters are used at every time step.

(Vanilla) Recurrent Neural Network



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

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

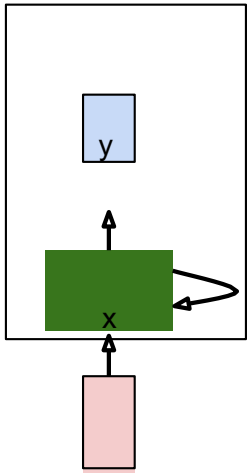
$$y_t = W_{hy} h_t$$

Example

Character-level language model example

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”

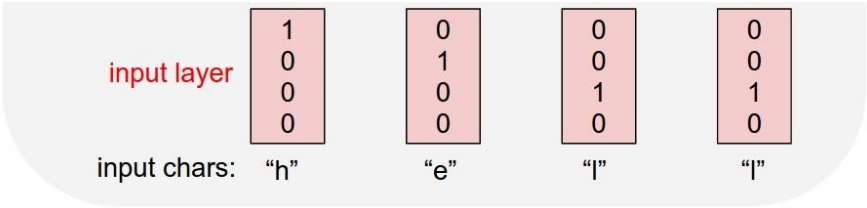


Example

Character-level language model example

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”



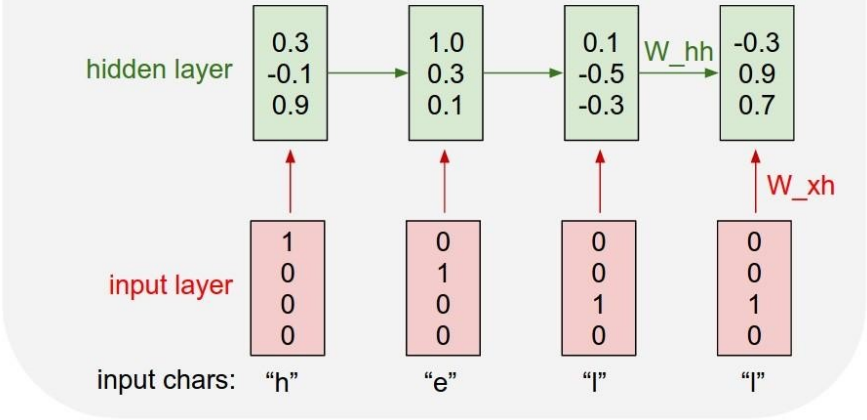
Example

Character-level language model example

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”

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

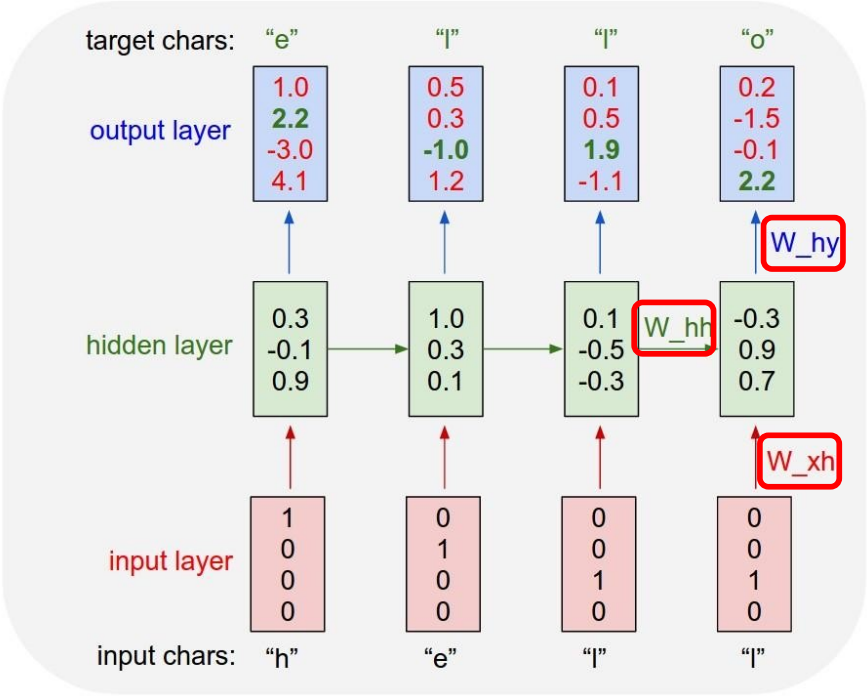


Example

Character-level language model example

Vocabulary:
[h,e,l,o]

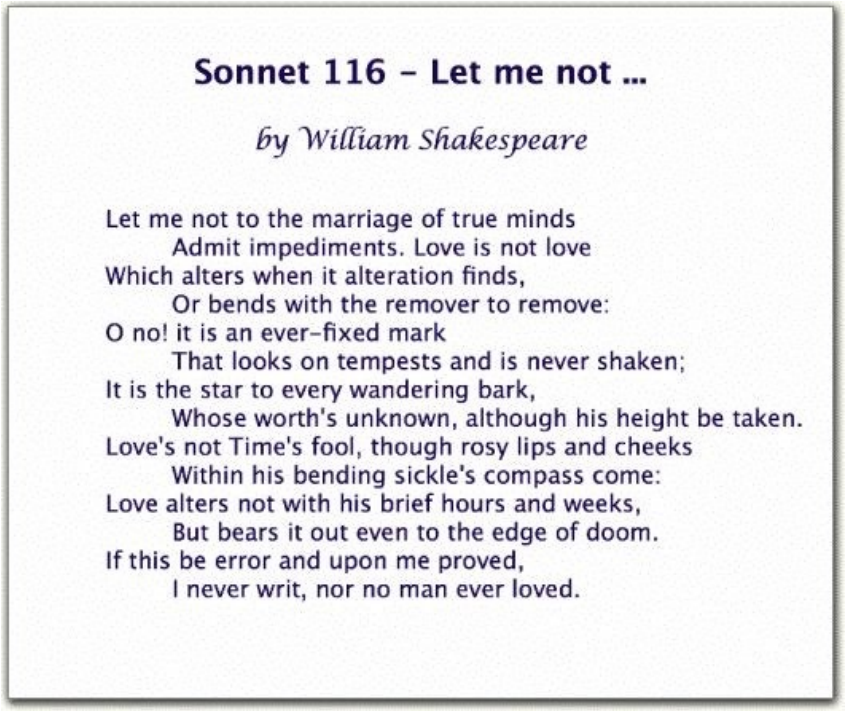
Example training sequence:
“hello”



Extensions

- Vanishing gradient problem makes it hard to model long sequences
  - Multiplying together many values between 0 and 1 (range of gradient of sigmoid, tanh)
- One solution: Use RELU
- Another solution: Use RNNs with gates
  - Adaptively decide how much of memory to keep
  - Gated Recurrent Units (GRUs), Long Short Term Memories (LSTMs)

Generating poetry with RNNS



Sonnet 116 - Let me not ...

by William Shakespeare

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

Generating poetry with RNNS

at first:

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

train more

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

train more

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

train more

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

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

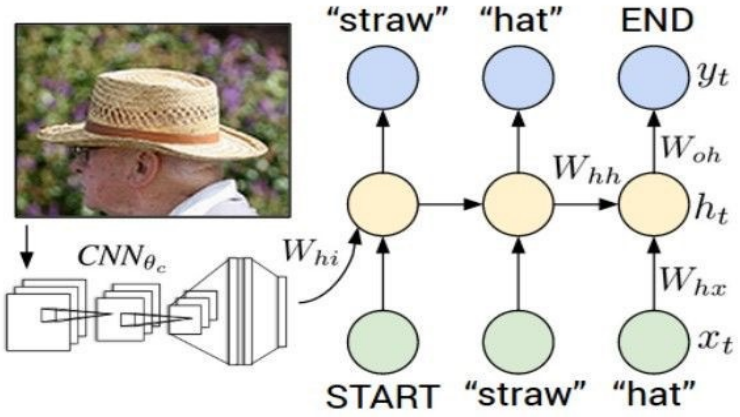
Andrej Karpathy

Generating poetry with RNNS

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

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

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

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

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

VIOLA:
I'll drink it.
```

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

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

Image Captioning

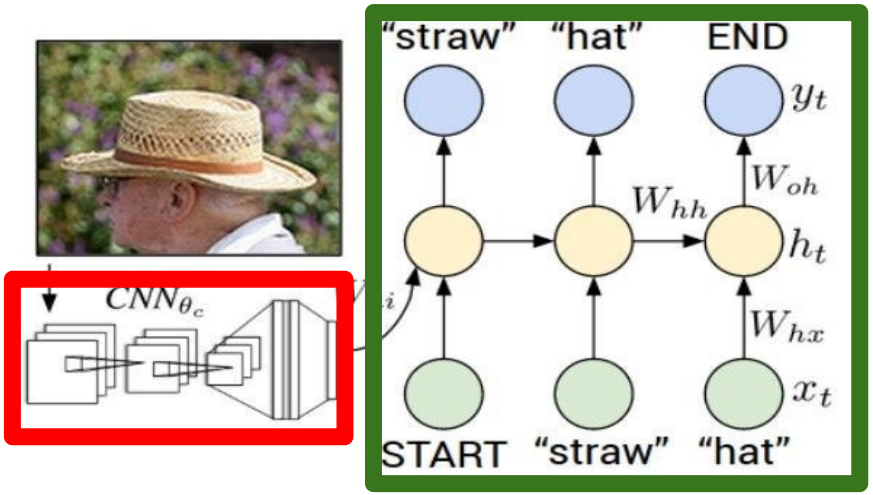


CVPR 2015:
Deep Visual-Semantic Alignments for Generating Image Descriptions, Karpathy and Fei-Fei
Show and Tell: A Neural Image Caption Generator, Vinyals et al.
Long-term Recurrent Convolutional Networks for Visual Recognition and Description, Donahue et al.
Learning a Recurrent Visual Representation for Image Caption Generation, Chen and Zitnick

Adapted from Andrej Karpathy

Image Captioning

Recurrent Neural Network

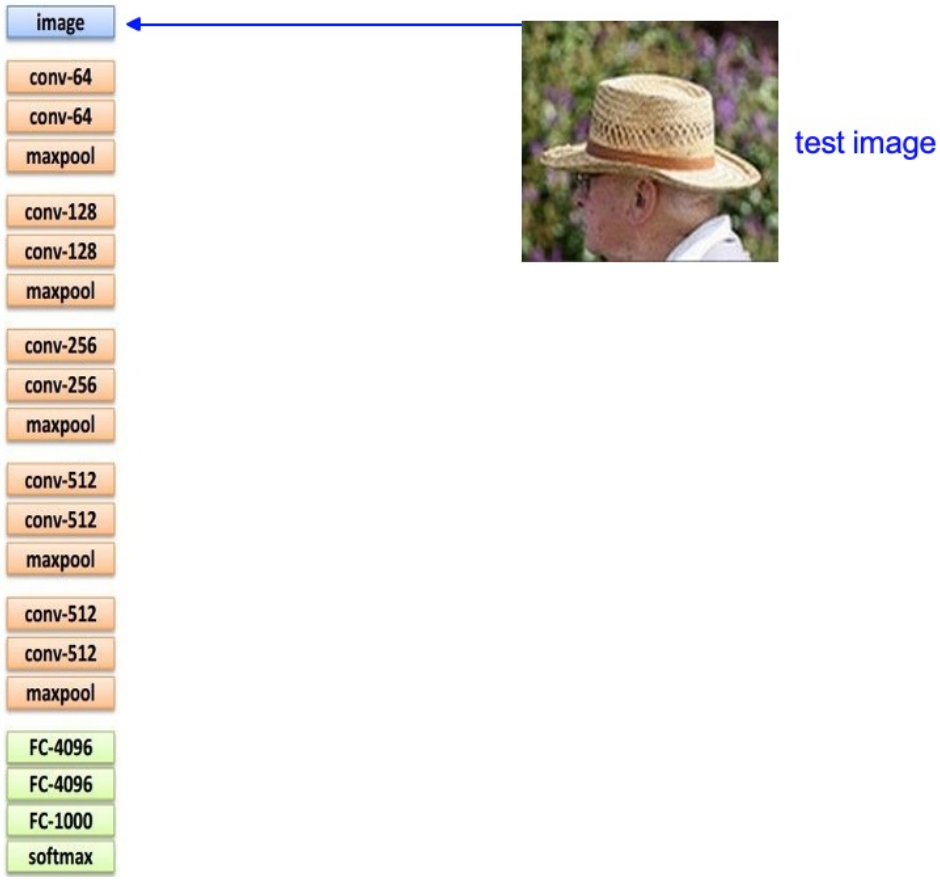


Convolutional Neural Network

Image Captioning

test image

Image Captioning



Andrej Karpathy

Image Captioning

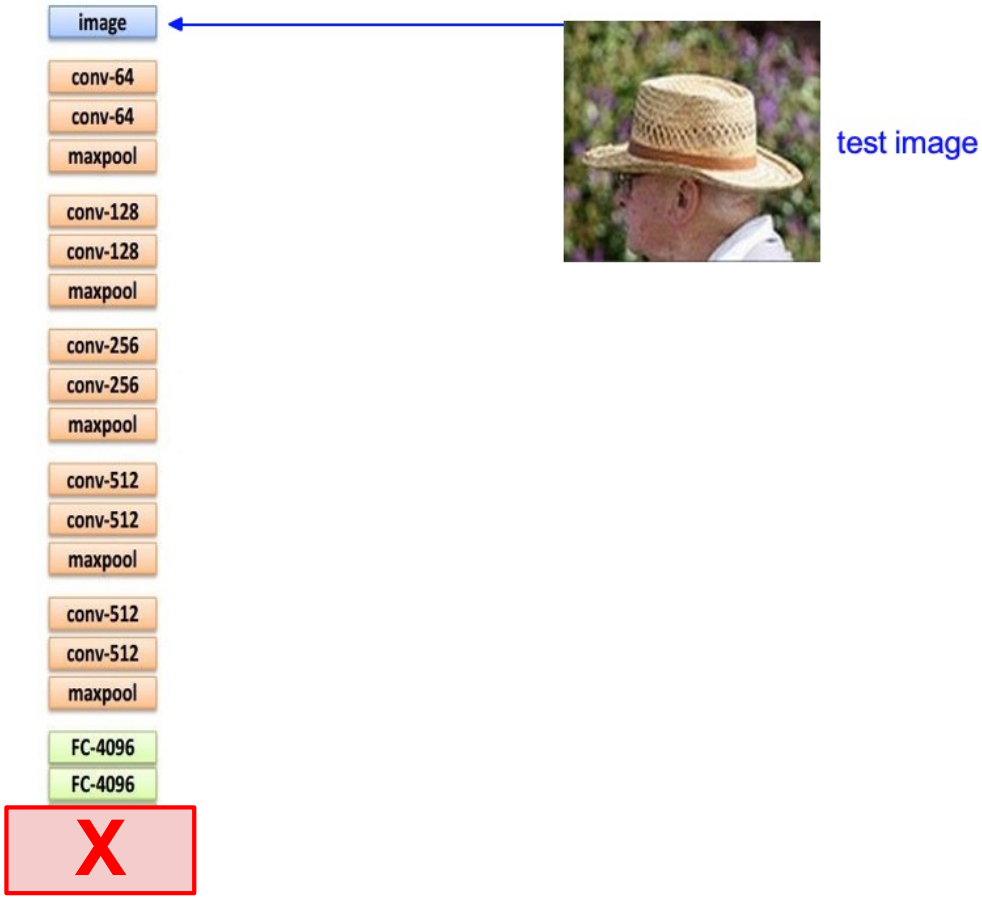
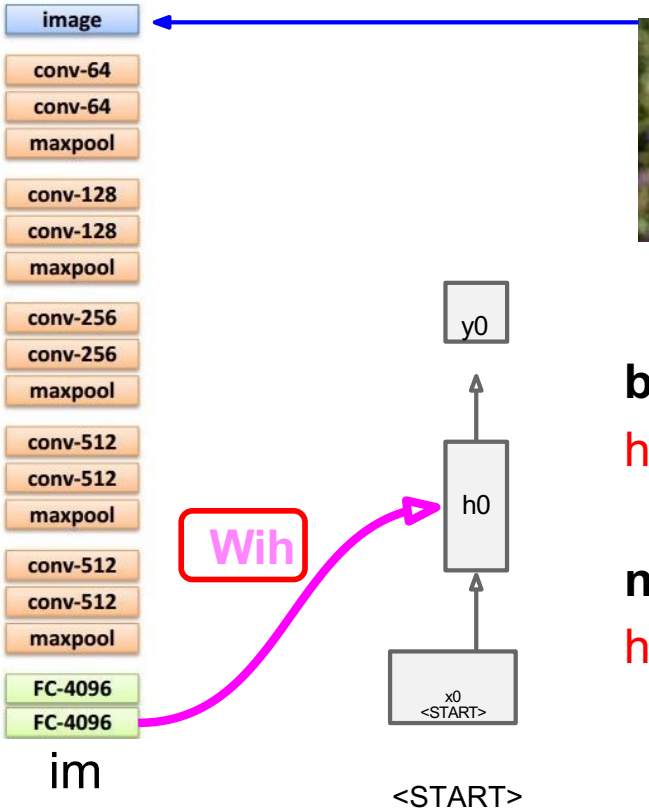


Image Captioning

Image Captioning



test image

before:

$h = \tanh(W_{xh} * x + W_{hh} * h)$

now:

$h = \tanh(W_{xh} * x + W_{hh} * h + W_{ih} * im)$

Image Captioning

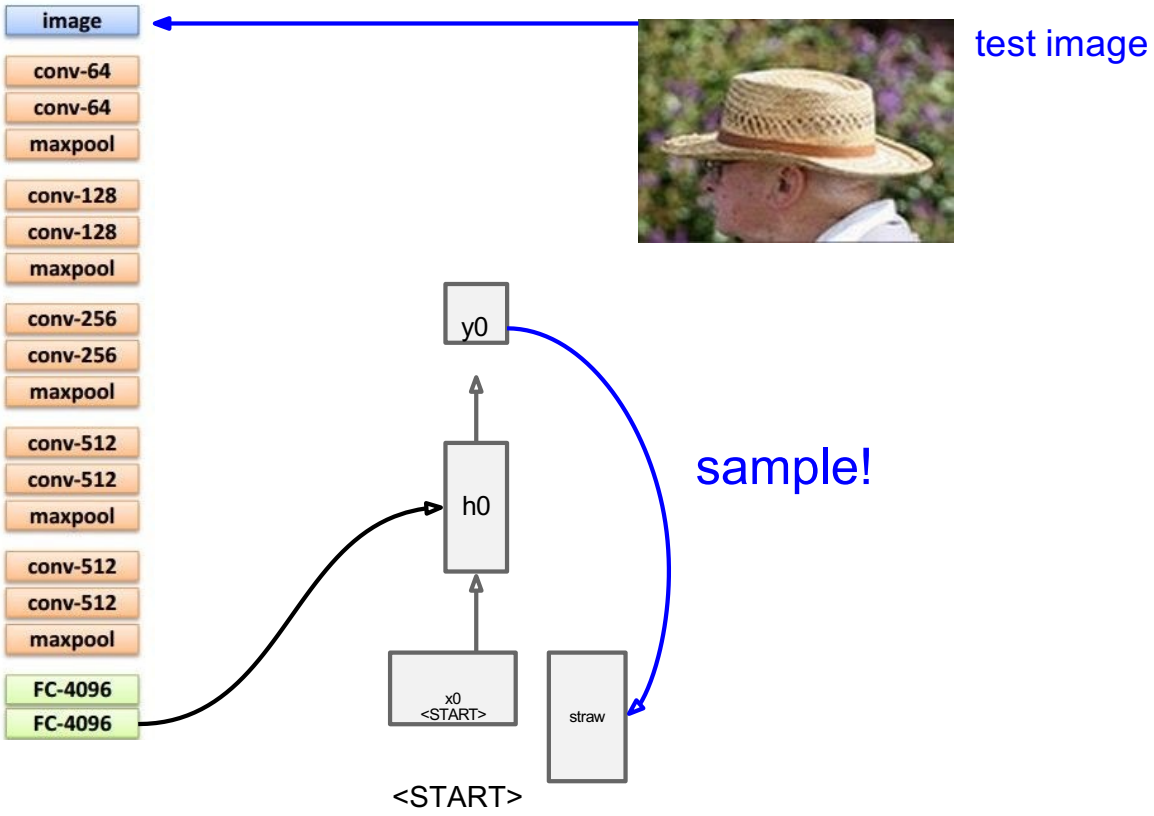


Image Captioning

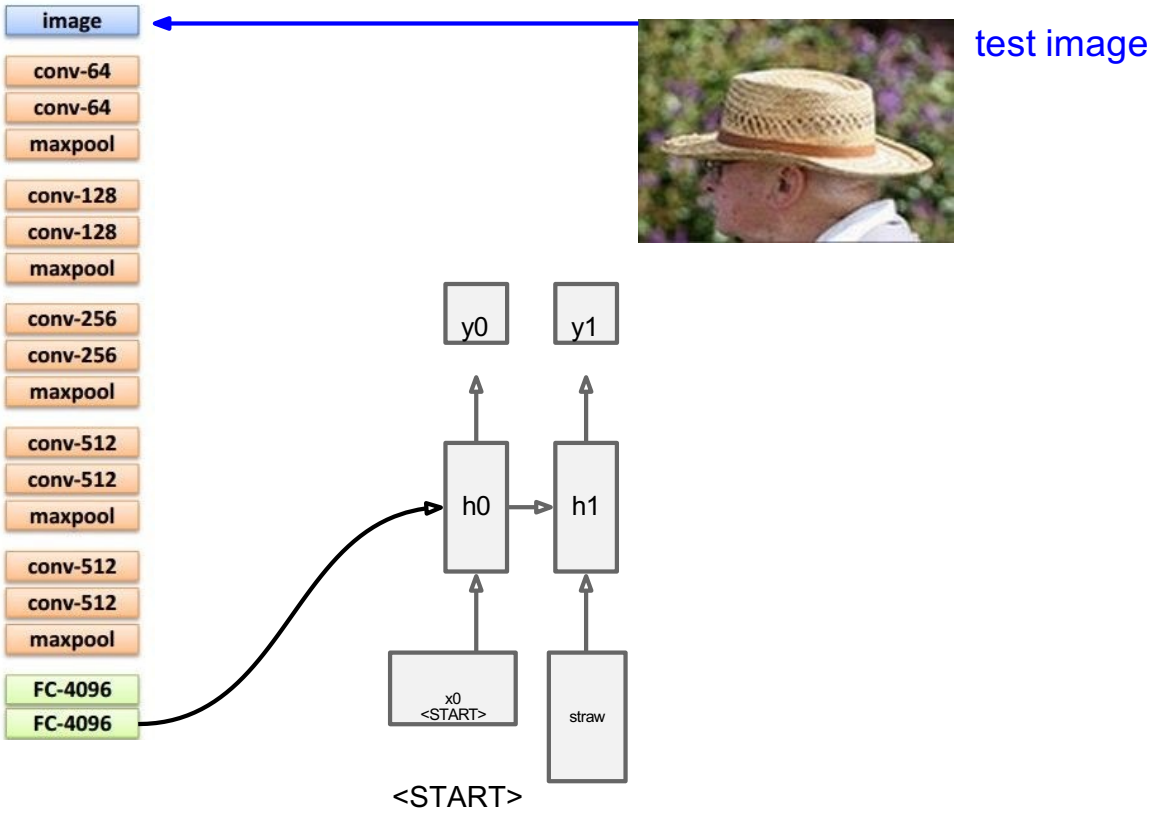


Image Captioning

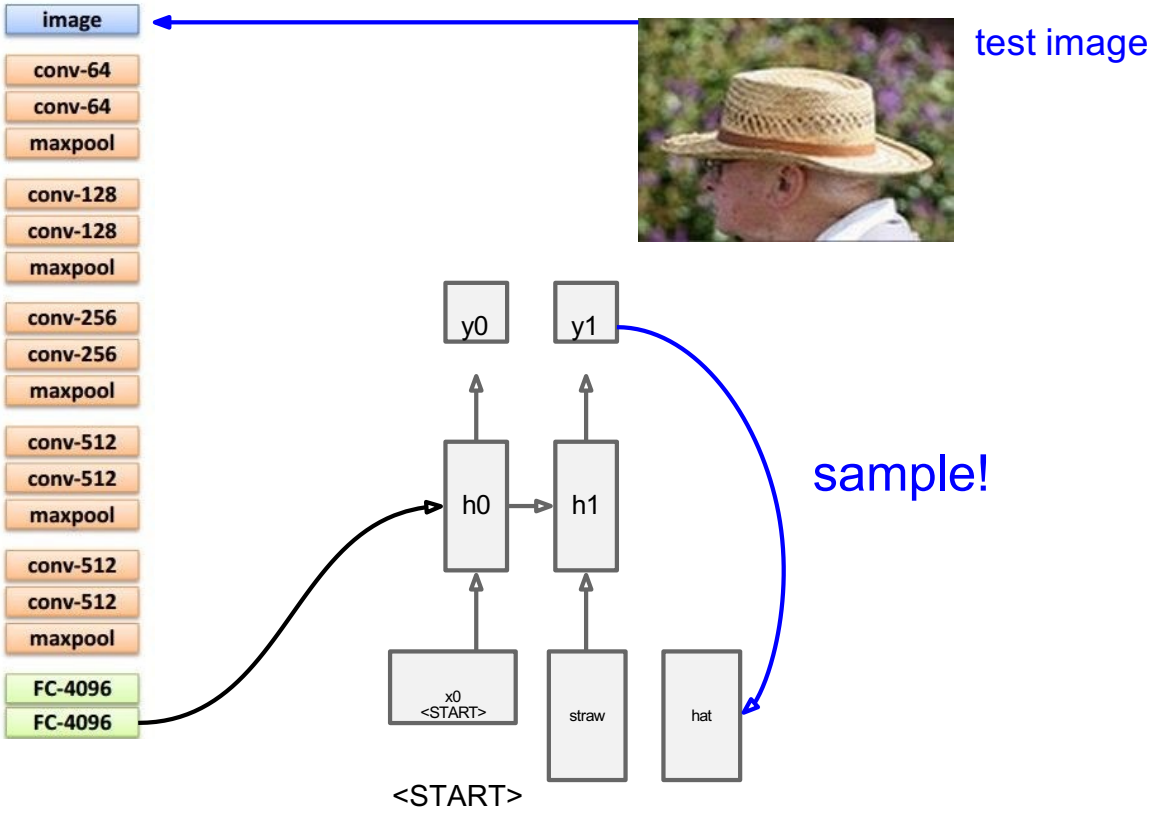


Image Captioning

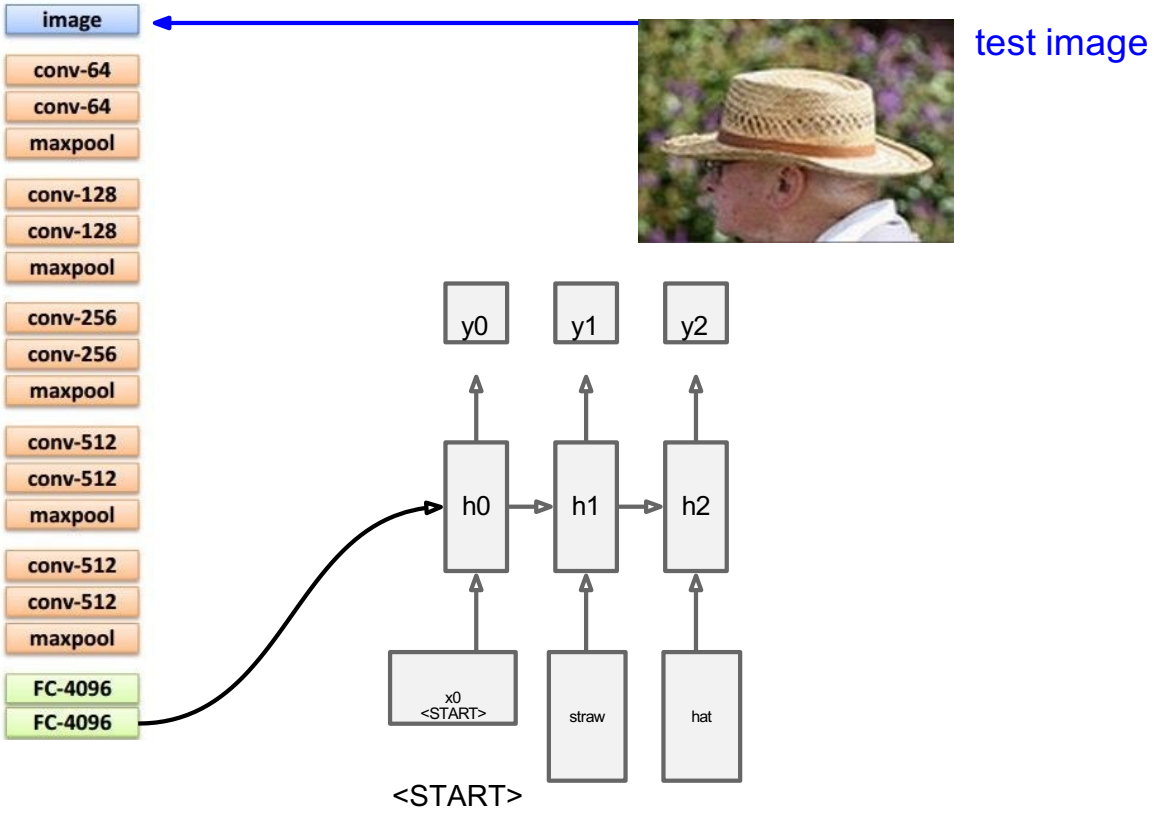
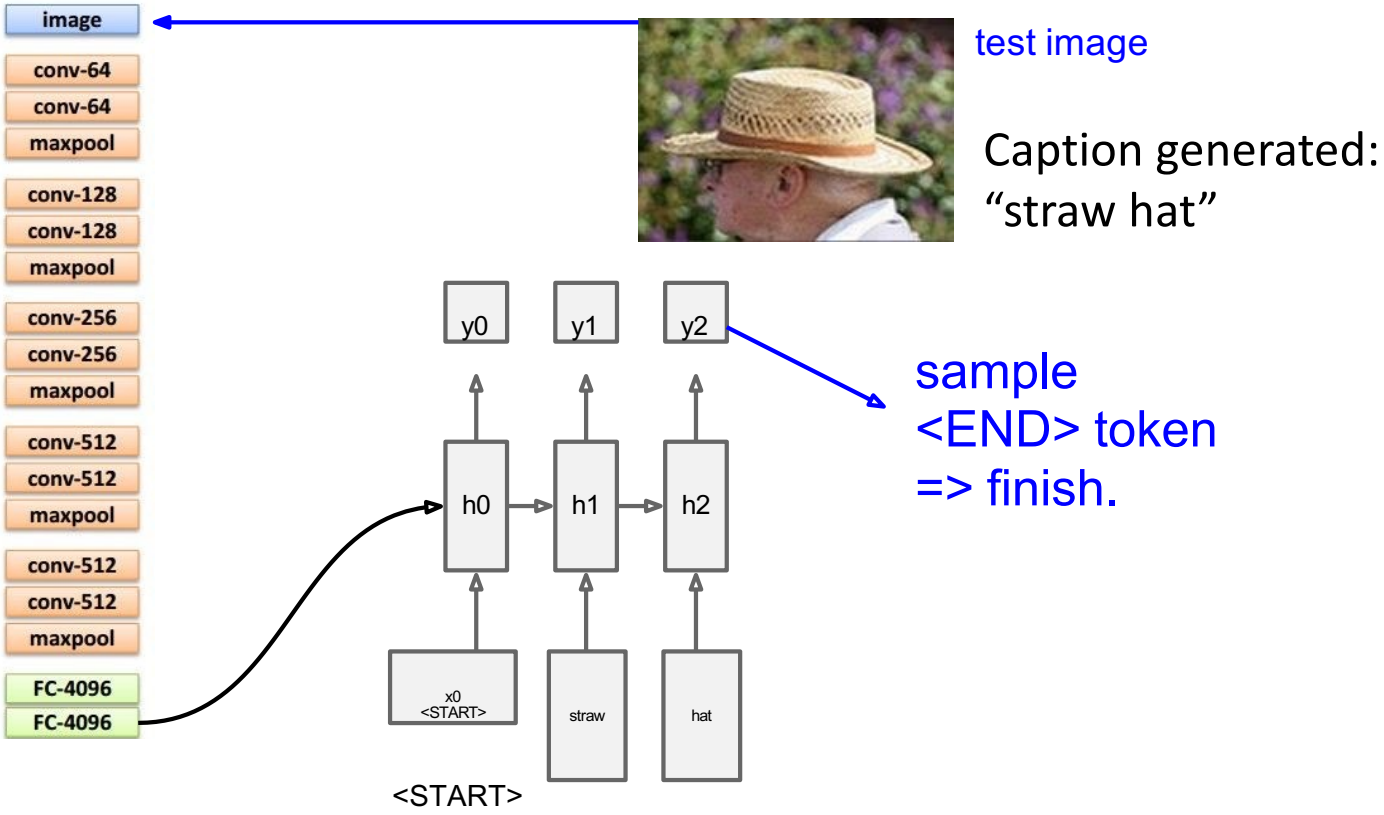


Image Captioning



Adapted from Andrej Karpathy

Image Captioning

"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."

"a young boy is holding a baseball bat."

"a cat is sitting on a couch with a remote control."

"a woman holding a teddy bear in front of a mirror."

"a horse is standing in the middle of a road."

Andrej Karpathy

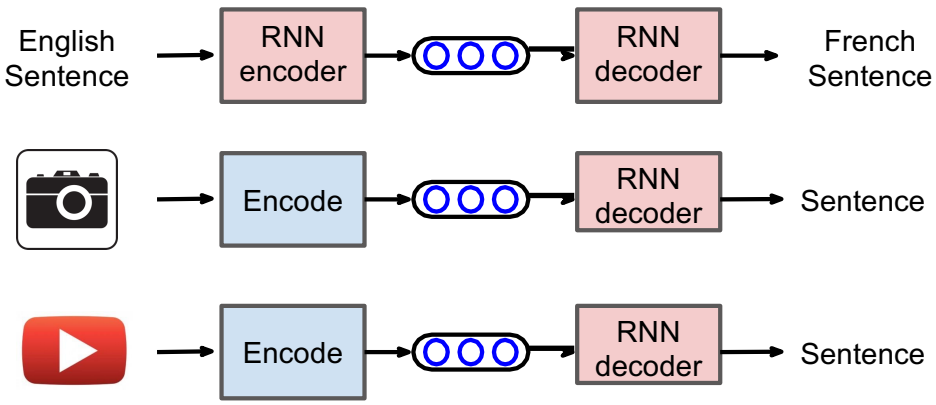
Video Captioning

Generate descriptions for events depicted in video clips

A monkey pulls a dog's tail and is chased by the dog.

Venugopalan et al., "Translating Videos to Natural Language using Deep Recurrent Neural Networks", NAACL-HTL 2015

Video Captioning

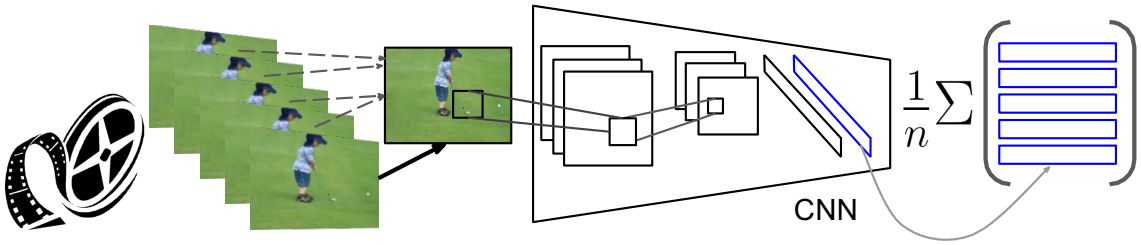


Key Insight:

Generate feature representation of the video and "decode" it to a sentence

Venugopalan et al., "Translating Videos to Natural Language using Deep Recurrent Neural Networks", NAACL-HTL 2015

Video Captioning



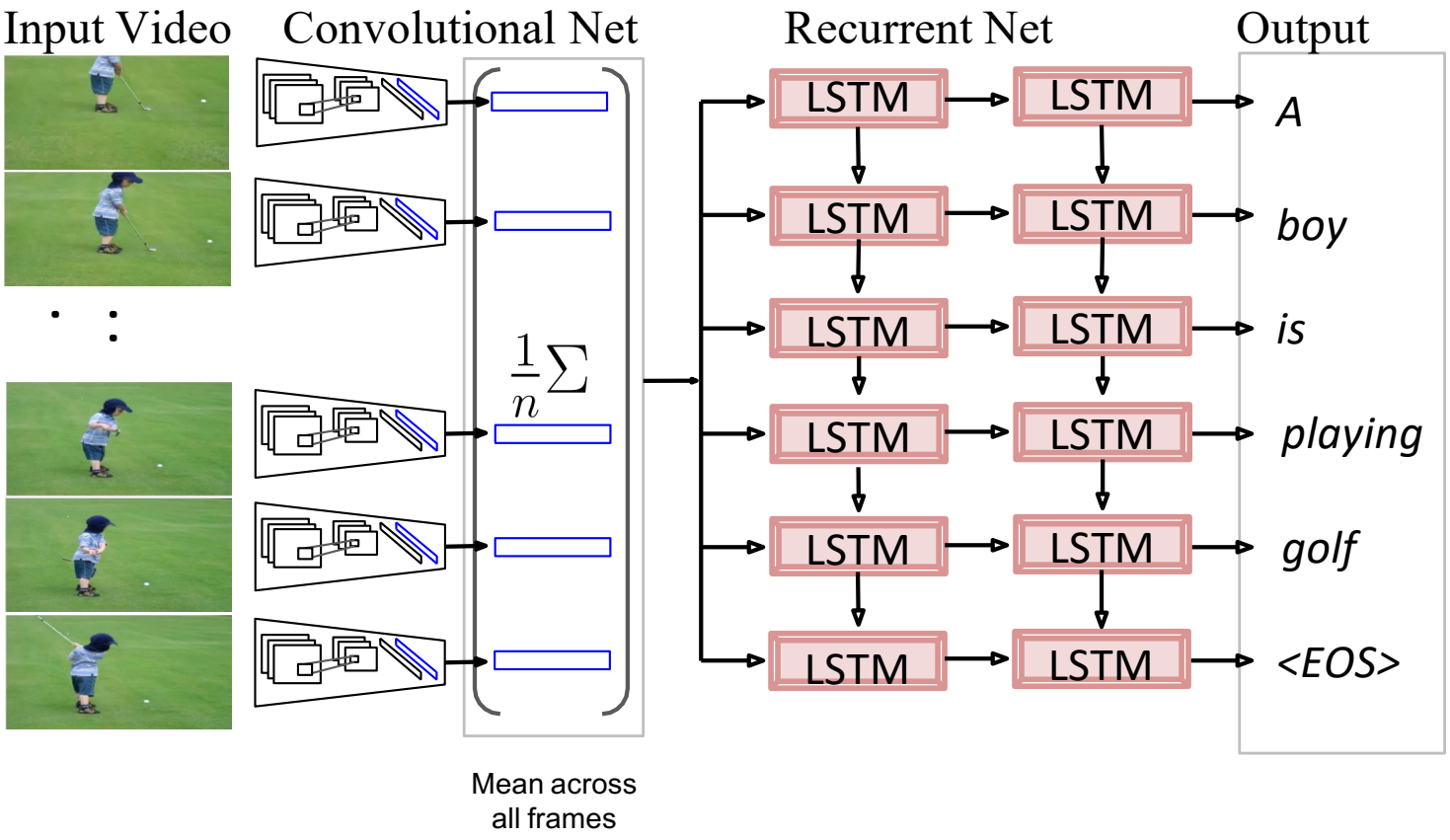
Input Video → Sample frames @1/10

Forward propagate
Output: "fc7" features
(activations before classification layer)

fc7: 4096 dimension
"feature vector"

Venugopalan et al., "Translating Videos to Natural Language using Deep Recurrent Neural Networks", NAACL-HTL 2015

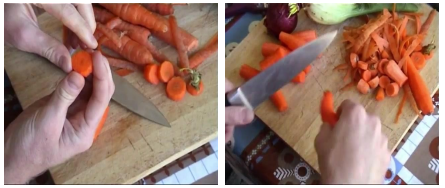
Video Captioning



Venugopalan et al., “Translating Videos to Natural Language using Deep Recurrent Neural Networks”, NAACL-HTL 2015

Video Captioning

FGM: A person is dancing with the person on the stage.
YT: A group of men are riding the forest.
I+V: **A group of people are dancing.**
GT: Many men and women are dancing in the street.



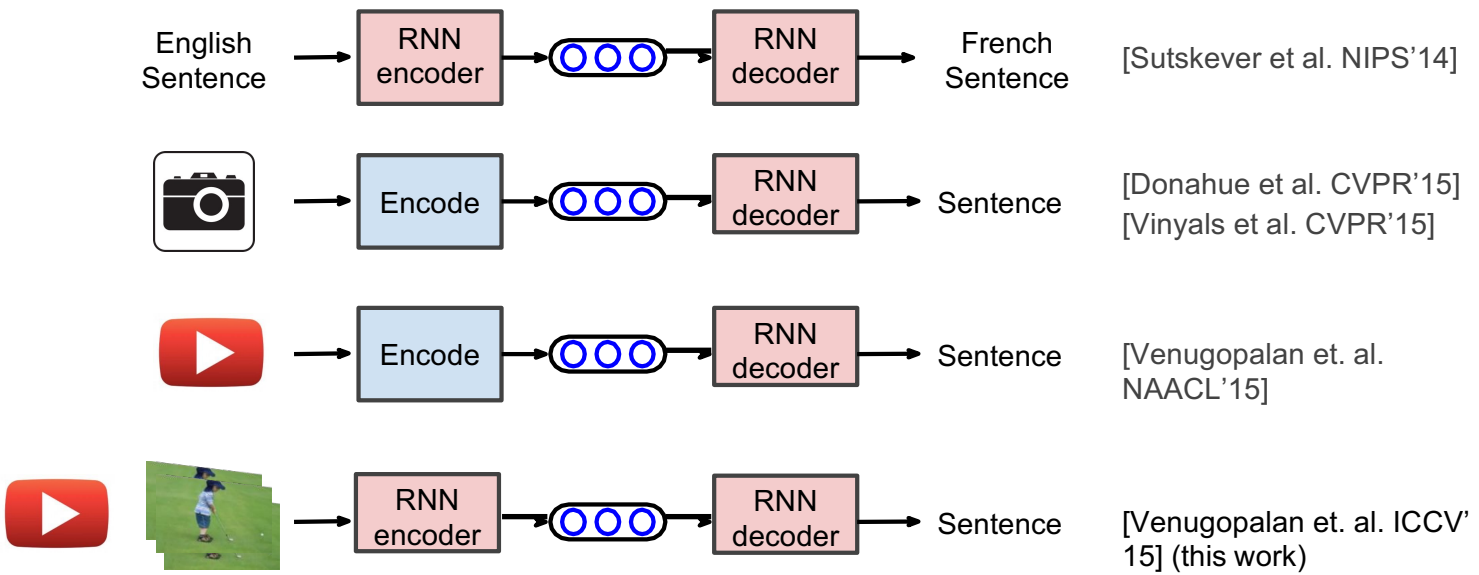
FGM: A person is cutting a potato in the kitchen.
YT: A man is slicing a tomato.
I+V: **A man is slicing a carrot.**
GT: A man is slicing carrots.

FGM: A person is walking with a person in the forest.
YT: A monkey is walking.
I+V: **A bear is eating a tree.**
GT: Two bear cubs are digging into dirt and plant matter at the base of a tree.

FGM: A person is riding a horse on the stage.
YT: A group of playing are playing in the ball.
I+V: **A basketball player is playing**.
GT: Dwayne wade does a fancy layup in an allstar game.

Venugopalan et al., “Translating Videos to Natural Language using Deep Recurrent Neural Networks”, NAACL-HTL 2015

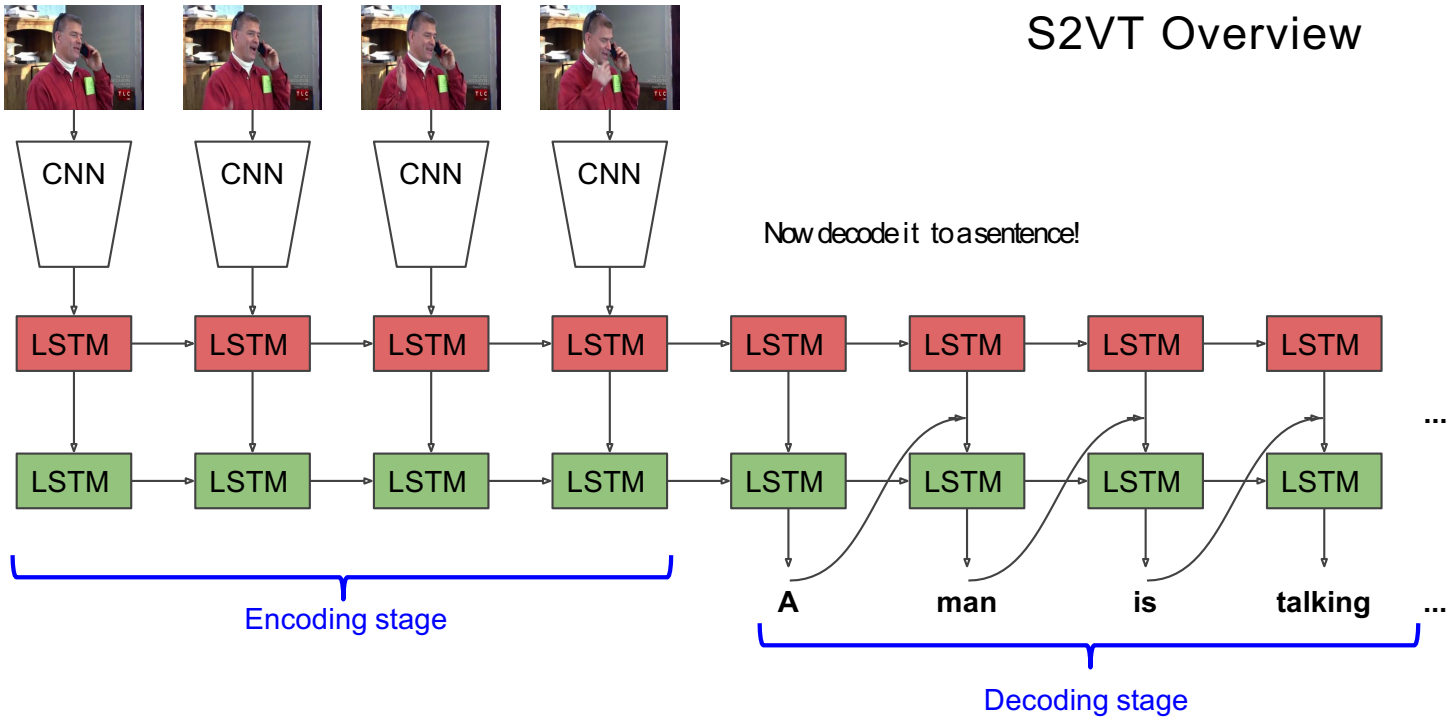
Video Captioning



3

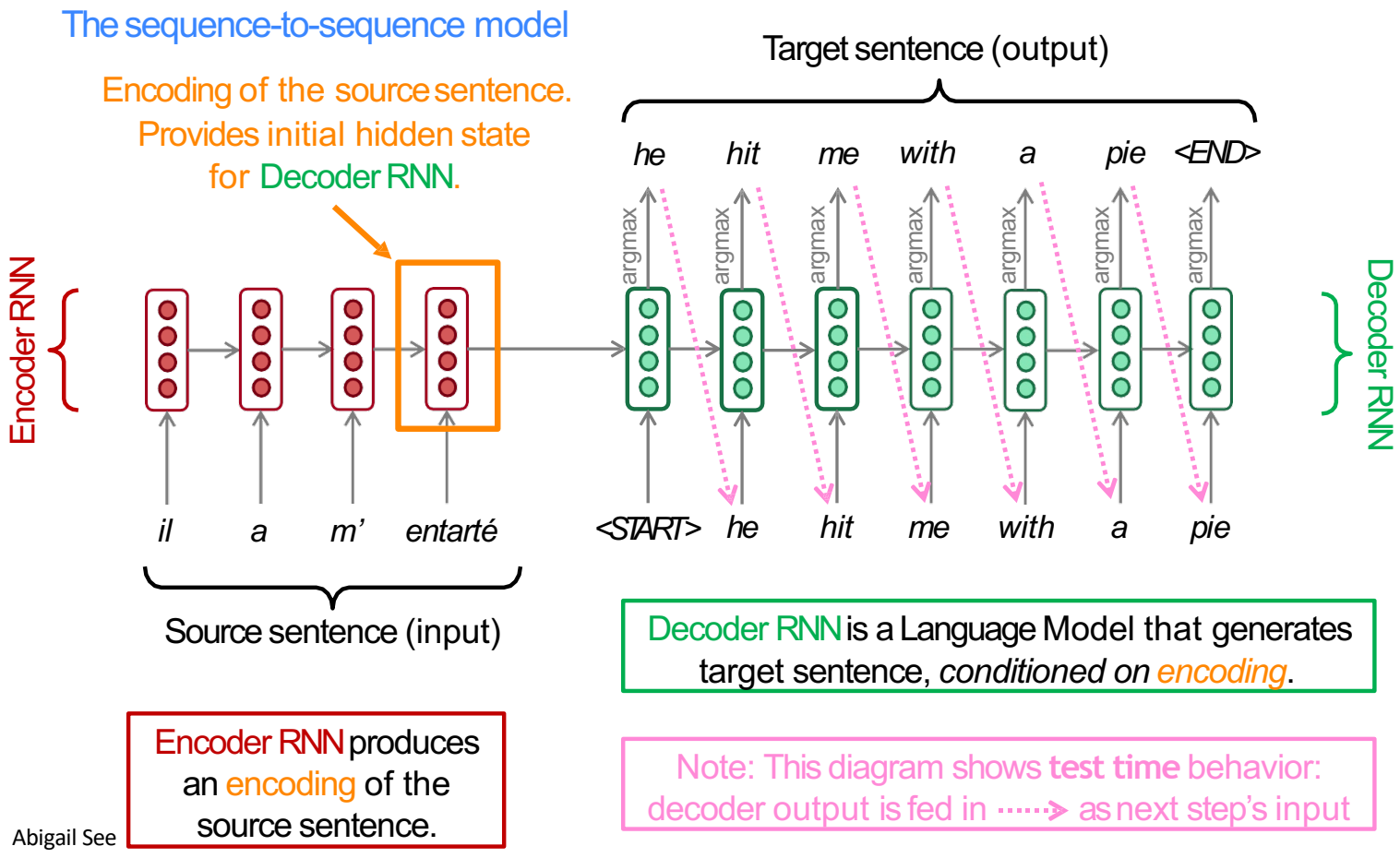
Venugopalan et al., "Sequence to Sequence - Video to Text", ICCV 2015

Video Captioning

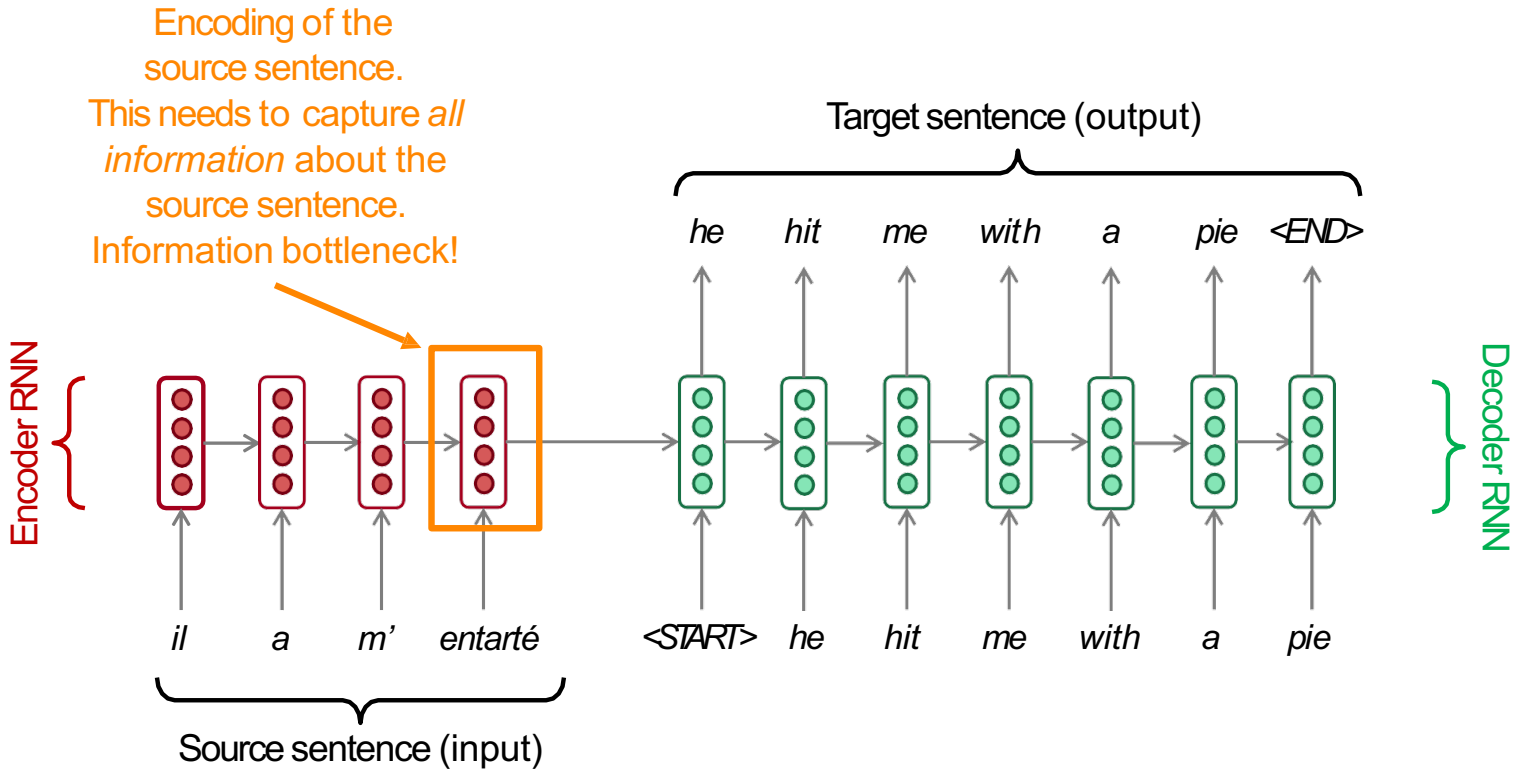


Venugopalan et al., “Sequence to Sequence - Video to Text”, ICCV 2015

Neural Machine Translation (NMT)



Sequence-to-sequence: the bottleneck problem



Abigail See

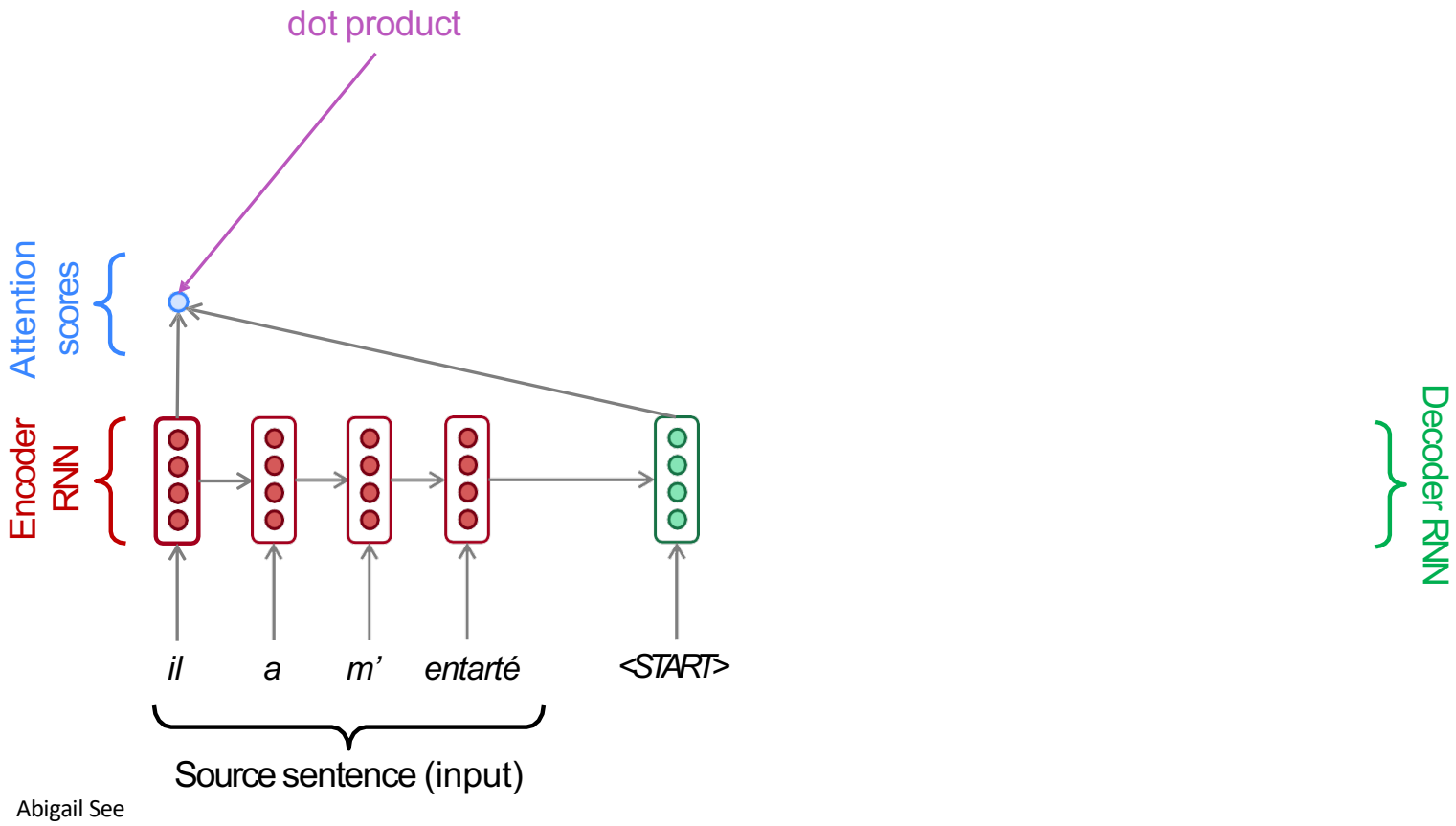
Attention

- Attention provides a solution to the bottleneck problem.

- Core idea: on each step of the decoder, use *direct connection to the encoder* to *focus on a particular part* of the source sequence

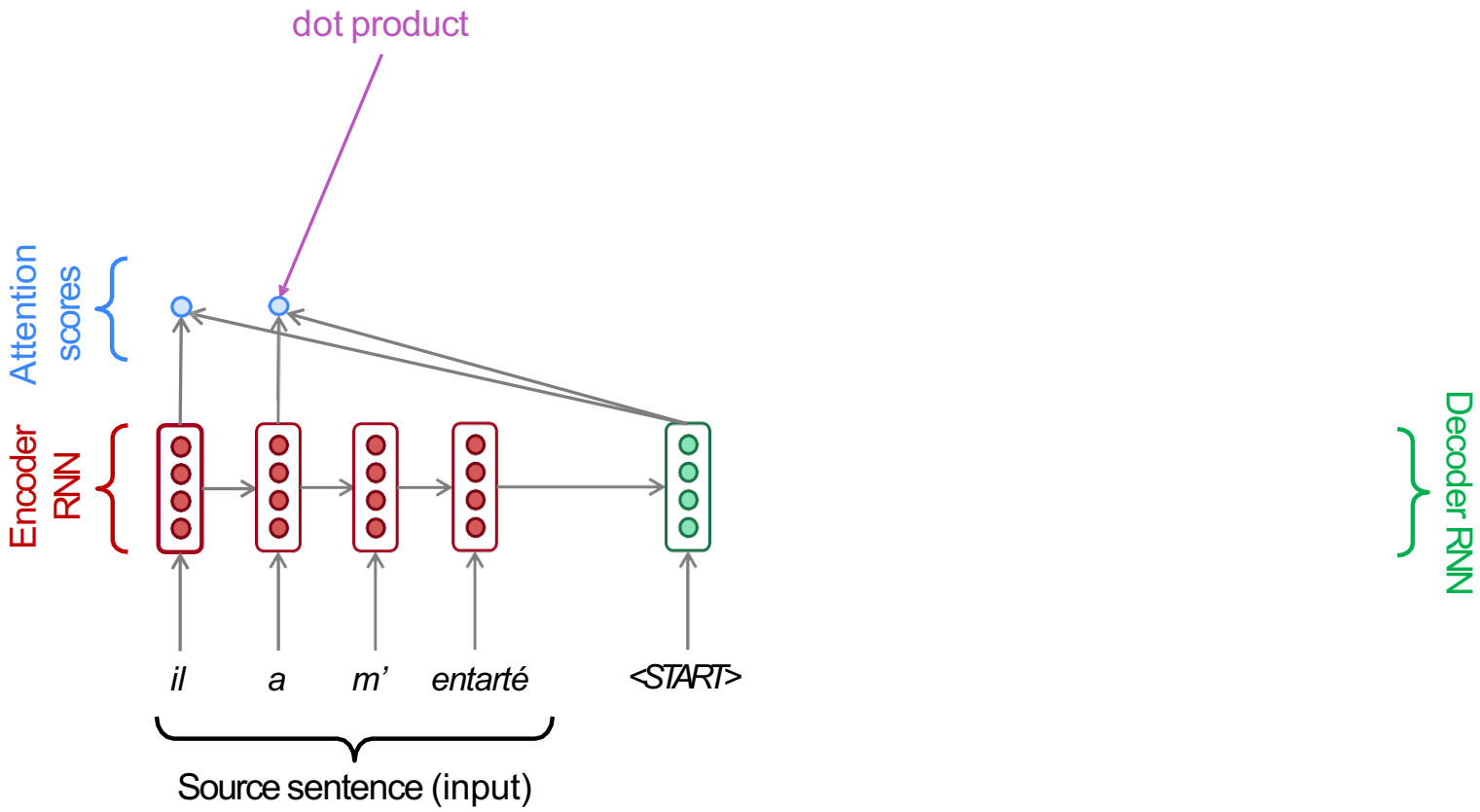
- First we will show via diagram (no equations), then we will show with equations

Sequence-to-sequence with attention



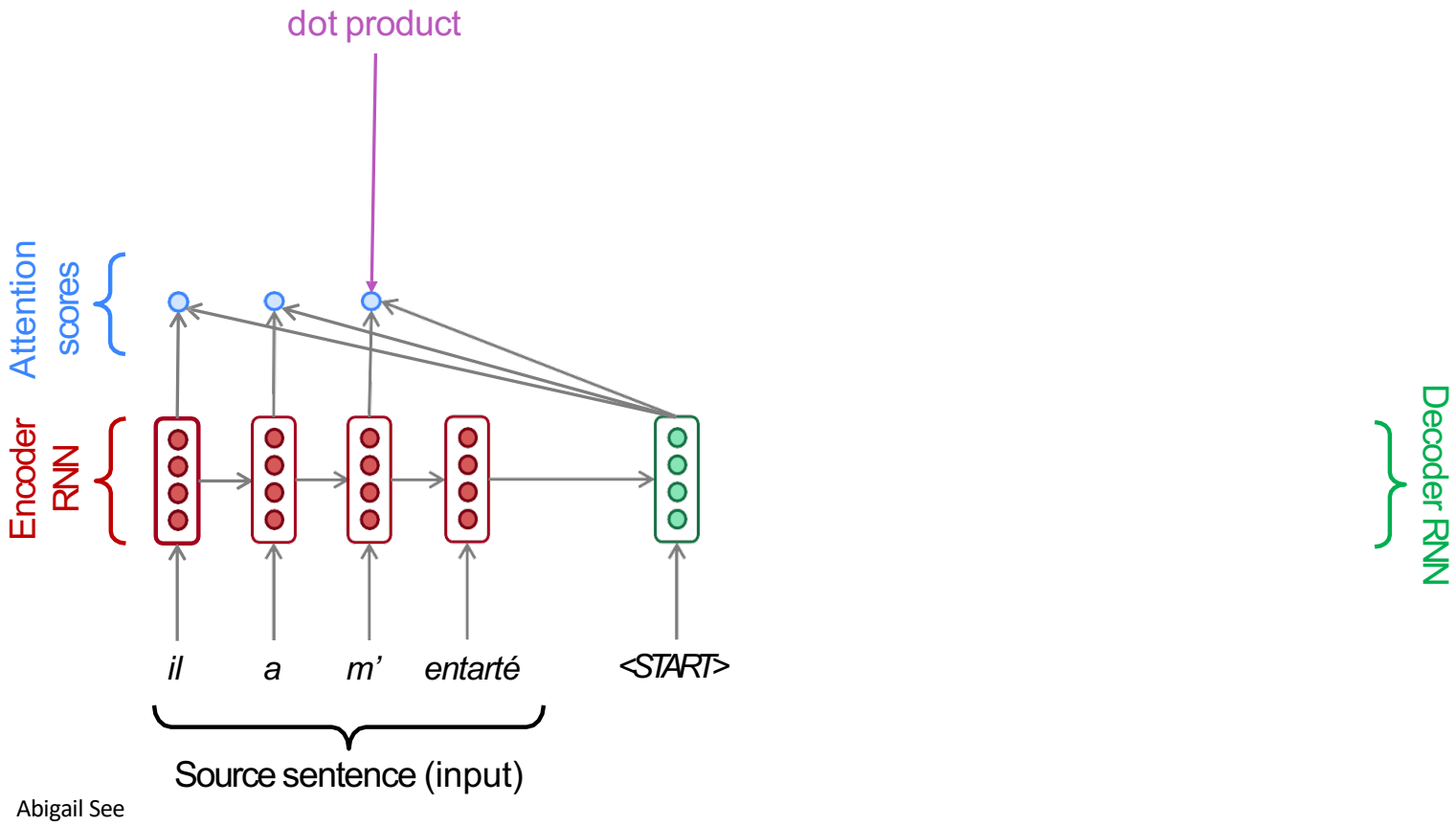
Abigail See

Sequence-to-sequence with attention



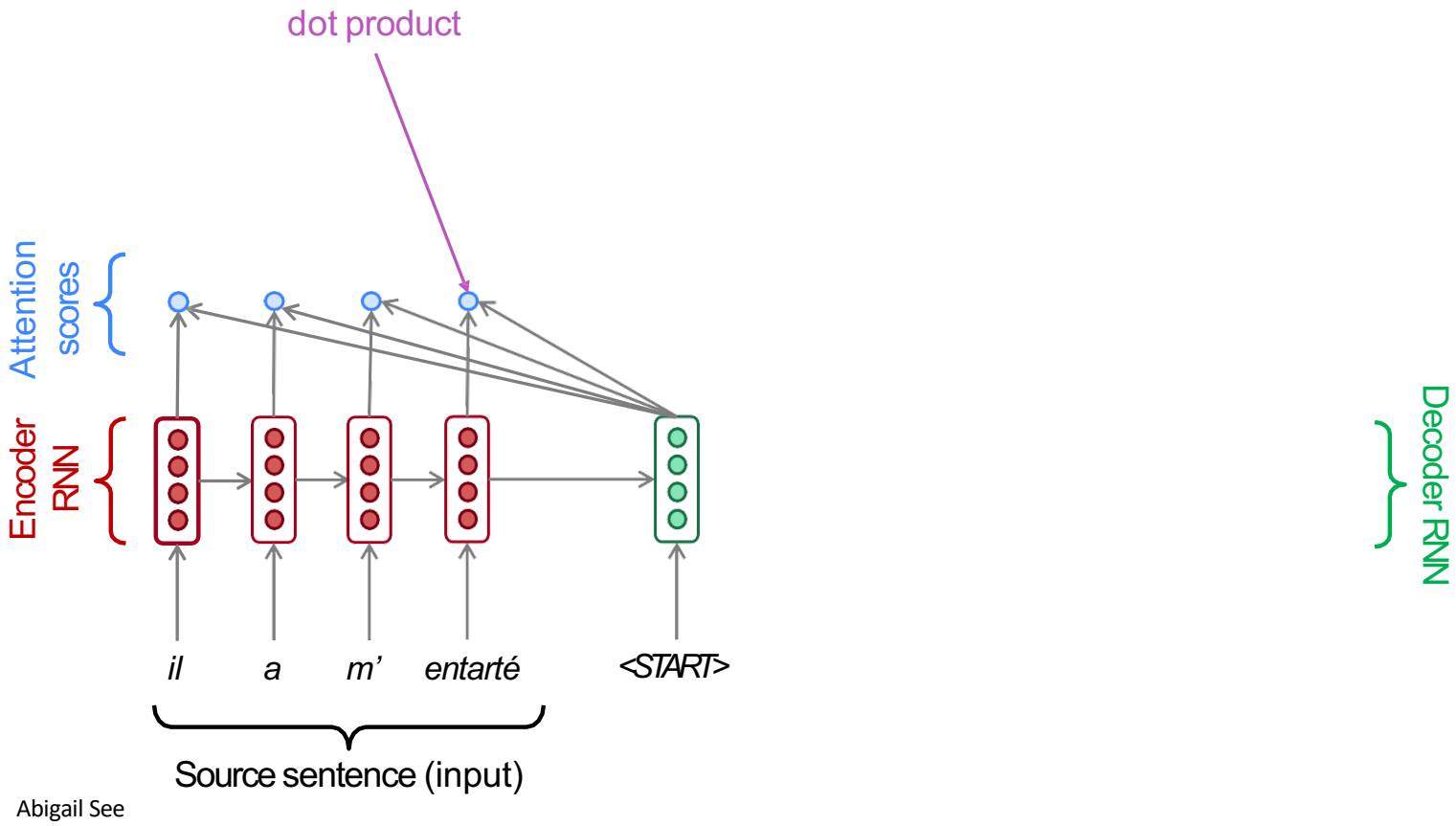
Abigail See

Sequence-to-sequence with attention



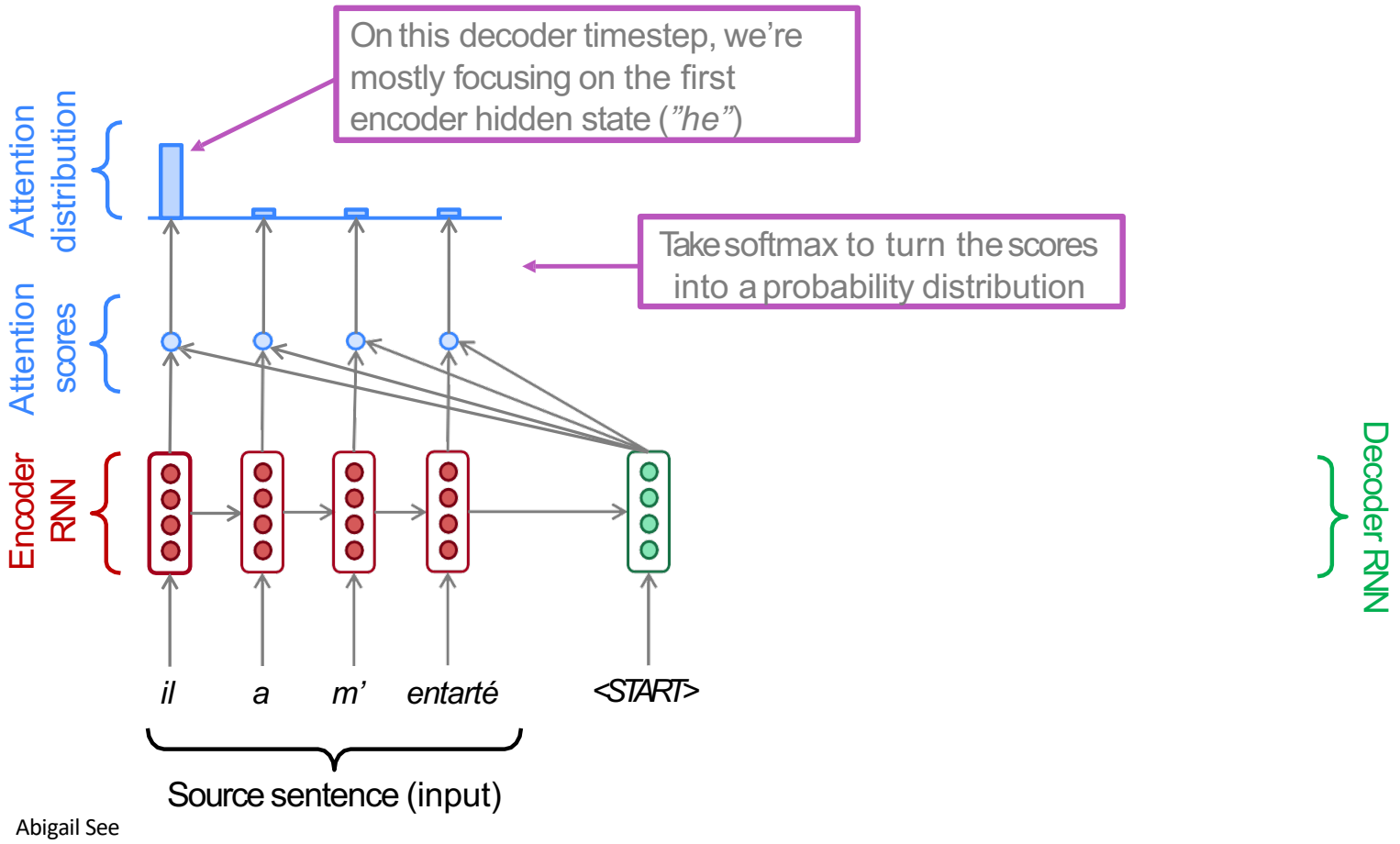
Abigail See

Sequence-to-sequence with attention



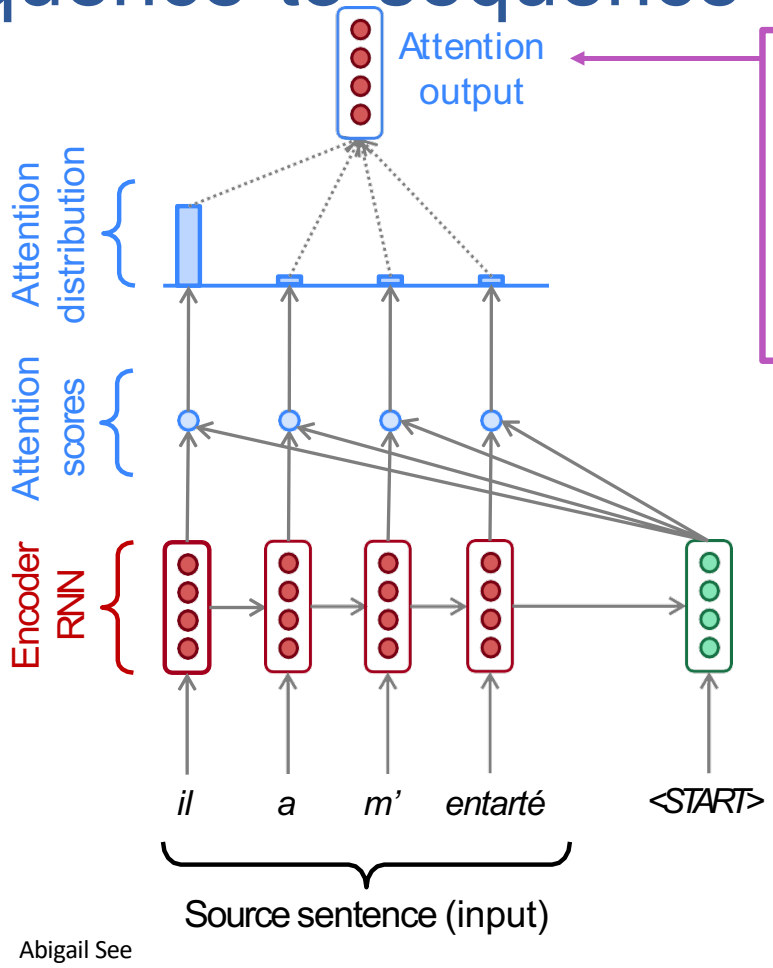
Abigail See

Sequence-to-sequence with attention



Abigail See

Sequence-to-sequence with attention



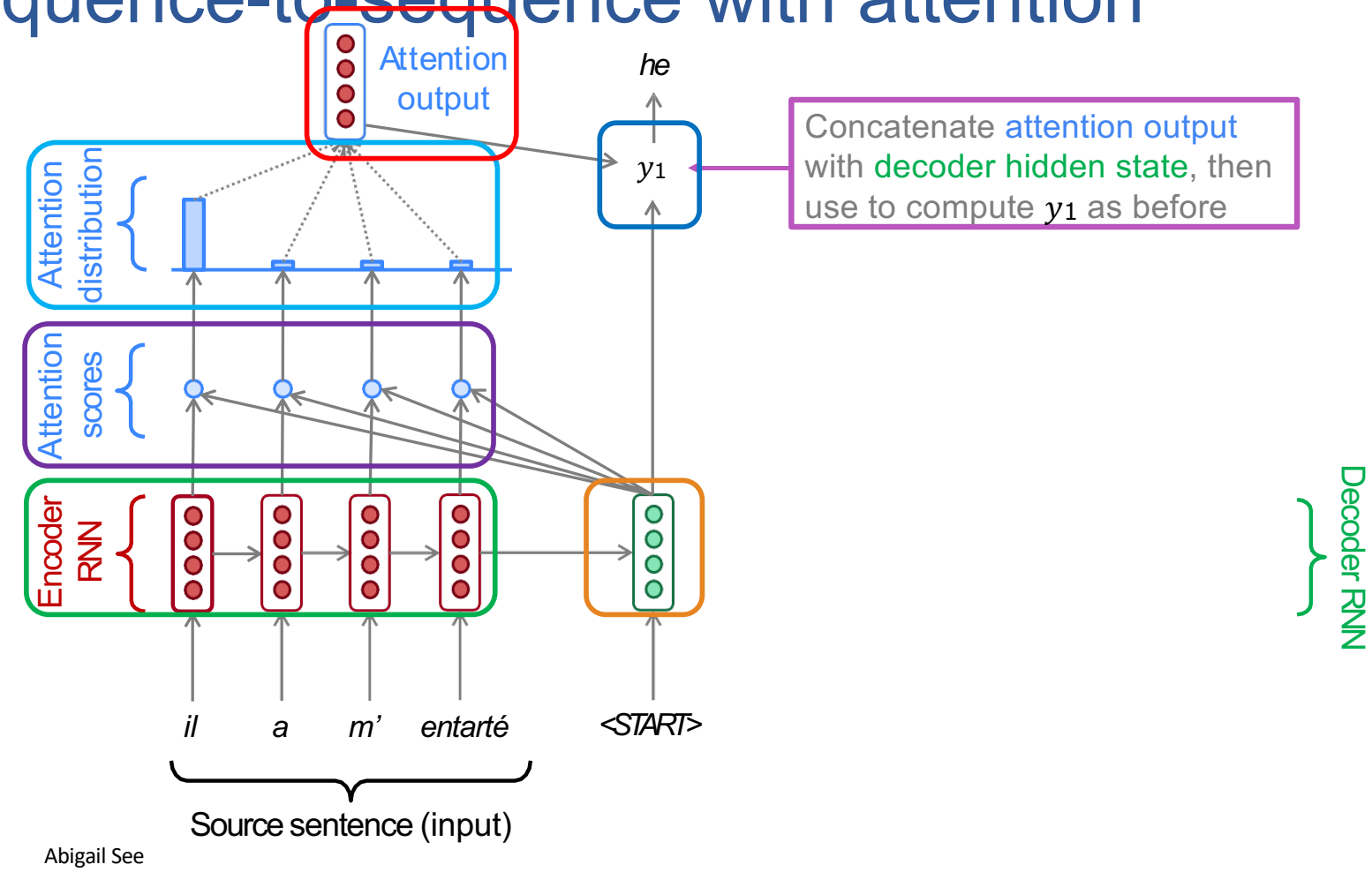
Use the attention distribution to take a **weighted sum** of the encoder hidden states.

The attention output mostly contains information from the hidden states that received high attention.

Decoder RNN

Abigail See

Sequence-to-sequence with attention



Abigail See

Attention in equations

- We have encoder hidden states $h_1, \ldots, h_N \in \mathbb{R}^h$
- On timestep *t*, we have decoder hidden state $s_t \in \mathbb{R}^h$
- We get the attention scores $\boldsymbol{e}^t$ for this step:

$$\boldsymbol{e}^t = [\boldsymbol{s}_t^T \boldsymbol{h}_1, \ldots, \boldsymbol{s}_t^T \boldsymbol{h}_N] \in \mathbb{R}^N$$

- We take softmax to get the attention distribution α^t for this step (this is a probability distribution and sums to 1)

$$\alpha^t = \text{softmax}(\boldsymbol{e}^t) \in \mathbb{R}^N$$

- We use α^t to take a weighted sum of the encoder hidden states to get the attention output $\boldsymbol{a}_t$

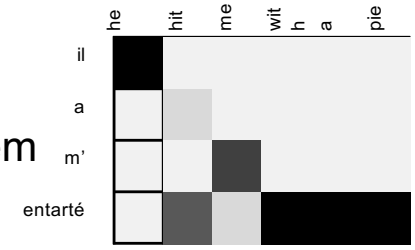
$$\boldsymbol{a}_t = \sum_{i=1}^{N} \alpha_i^t \boldsymbol{h}_i \in \mathbb{R}^h$$

- Finally we concatenate the attention output $\boldsymbol{a}_t$ with the decoder hidden state s_t and proceed as in the non-attention seq2seq model

$$[\boldsymbol{a}_t; \boldsymbol{s}_t] \in \mathbb{R}^{2h}$$

Attention is great!

- Attention significantly improves NMT performance
  - It's very useful to allow decoder to focus on certain parts of the source
- Attention solves the bottleneck problem
  - Attention allows decoder to look directly at source; bypass bottleneck
- Attention helps with vanishing gradient problem
  - Provides shortcut to faraway states
- Attention provides some interpretability
  - By inspecting attention distribution, we can see what the decoder was focusing on
  - We get (soft) alignment for free!
  - This is cool because we never explicitly trained an alignment system
  - The network just learned alignment by itself



Attention is a general deep-learning technique

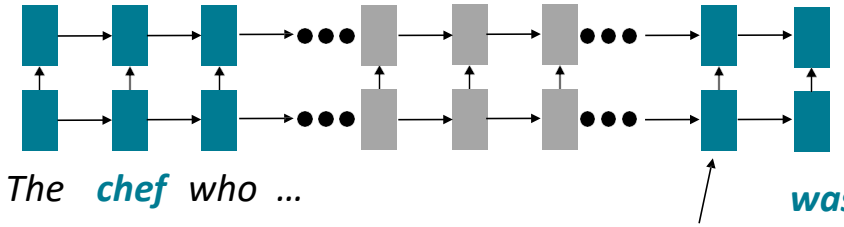
- We've seen that attention is a great way to improve the sequence-to-sequence model for Machine Translation.
- However: You can use attention in many architectures (not just seq2seq) and many tasks (not just MT)

> - More general definition of attention:
>   - Given a set of vector *values*, and a vector *query*, attention is a technique to compute a weighted sum of the values, dependent on the query.

- We sometimes say that the query *attends to* the values.
- For example, in seq2seq + attention model, each decoder hidden state (query) *attends to* all encoder hidden states (values).

Issues with recurrent models: Linear interaction distance

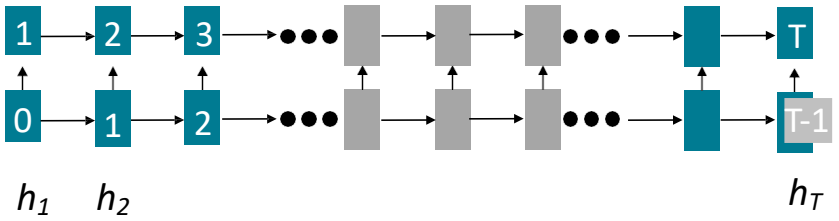
- **O(sequence length)** steps for distant word pairs to interact means:
  - Hard to learn long-distance dependencies (because gradient problems!)
  - Linear order of words is "baked in"; not necessarily the right way to think about sentences…



Adapted from John Hewitt

Issues with recurrent models: Lack of parallelization

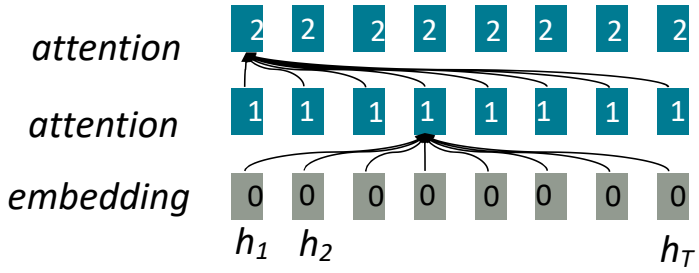
- Forward and backward passes have **O(sequence length)** unparallelizable operations
  - GPUs can perform a bunch of independent computations at once!
  - But future RNN hidden states can't be computed in full before past RNN hidden states have been computed
  - Inhibits training on very large datasets!



Numbers indicate min # of steps before a state can be computed

If not recurrence, then what? How about attention?

- **Attention** treats each word's representation as a **query** to access and incorporate information from **a set of values.**
  - We saw attention from the **decoder** to the **encoder**; next we'll think about attention **within a single sentence**.
    - If **attention** gives us access to any state… maybe we can just use attention and don't need the RNN?
- Number of unparallelizable operations not tied to sequence length.
- All words interact at every layer!



All words attend to all words in previous layer; most arrows here are omitted

Adapted from John Hewitt

Self-Attention

- Attention operates on **queries**, **keys**, and **values**.
  - We have some **queries** $q_1, q_2, \ldots, q_T$. Each query is $q_i \in \mathbb{R}^d$
  - We have some **keys** $k_1, k_2, \ldots, k_T$. Each key is $k_i \in \mathbb{R}^d$
  - We have some **values** $v_1, v_2, \ldots, v_T$. Each value is $v_i \in \mathbb{R}^d$

The number of queries can differ from the number of keys and values in practice.

- In **self-attention**, the queries, keys, and values are drawn from the same source.
  - For example, if the output of the previous layer is $x_1, \ldots, x_T$, (one vec per word) we could let $v_i = k_i = q_i = x_i$ (that is, use the same vectors for all of them!)
- The (dot product) self-attention operation is as follows:

$$e_{ij} = q_i^\top k_j \qquad \alpha_{ij} = \frac{\exp(e_{ij})}{\Sigma_{j'} \exp(e_{ij'})} \qquad \text{output}_i = \Sigma_j \alpha_{ij} v_j$$

Compute **key-query** affinities

Compute attention weights from affinities (softmax)

Compute outputs as weighted sum of **values**

Barriers and solutions for Self-Attention as a building block

Barriers

- Doesn't have an inherent notion of order!

Solutions

Fixing the first self-attention problem: Sequence order

- Since self-attention doesn't build in order information, we need to encode the order of the sentence in our keys, queries, and values.
- Consider representing each **sequence index** as a **vector**

$$p_i \in \mathbb{R}^d, \text{ for } i \in \{1,2,\dots,T\} \text{ are position vectors}$$

- Don't worry about what the p_i are made of yet!
- Easy to incorporate this info into our self-attention block: just add the p_i to our inputs!
- Let $v_i{}', k_i{}', q_i{}'$ be our old values, keys, and queries.

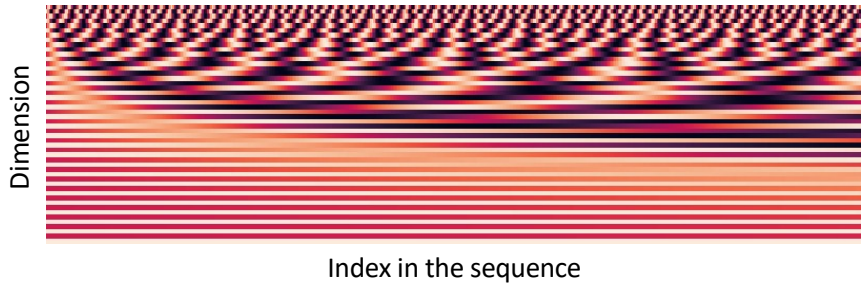
$$v_i = v_i{}' + p_i$$
$$q_i = q_i{}' + p_i$$
$$k_i = k_i{}' + p_i$$

In deep self-attention networks, we do this at the first layer! You could concatenate them as well, but people mostly just add…

Position representation vectors through sinusoids

- **Sinusoidal position representations:** concatenate sinusoidal functions of varying periods:

$$p_i = \begin{pmatrix} \sin(i/10000^{2*1/d}) \\ \cos(i/10000^{2*1/d}) \\ \vdots \\ \sin(i/10000^{2*\frac{d}{2}/d}) \\ \cos(i/10000^{2*\frac{d}{2}/d}) \end{pmatrix}$$



- Periodicity indicates that maybe "absolute position" isn't as important

Image: https://timodenk.com/blog/linear-relationships-in-the-transformers-positional-encoding/

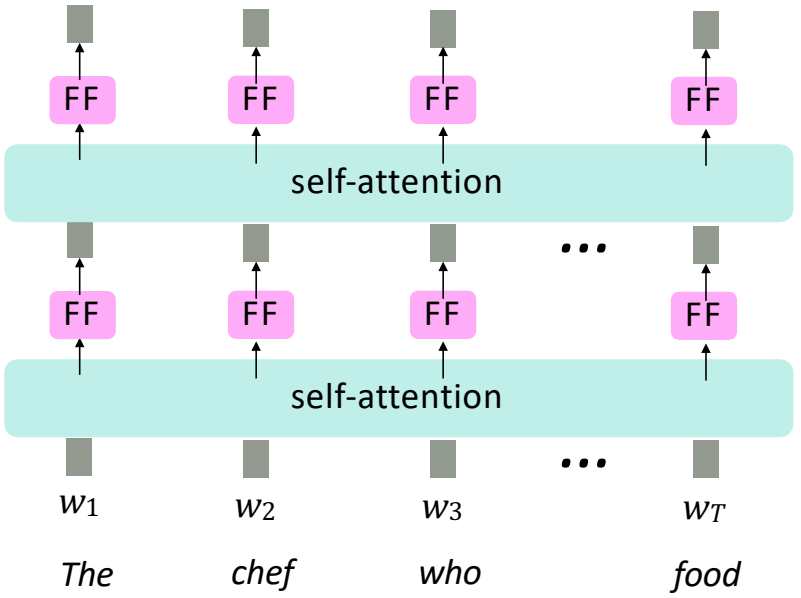
Barriers and solutions for Self-Attention as a building block

Barriers	Solutions
• Doesn't have an inherent notion of order!	• Add position representations to the inputs
• No nonlinearities for deep learning! It's all just weighted averages	

Adding nonlinearities in self-attention

- Note that there are no elementwise nonlinearities in self-attention; stacking more self-attention layers just re-averages **value** vectors
- Easy fix: add a **feed-forward network** to post-process each output vector.

$$m_i = MLP(\text{output}_i)$$
$$= W_2 * \mathbf{ReLU}(W_1 \times \text{output}_i + b_1) + b_2$$



Intuition: the FF network processes the result of attention

Barriers and solutions for Self-Attention as a building block

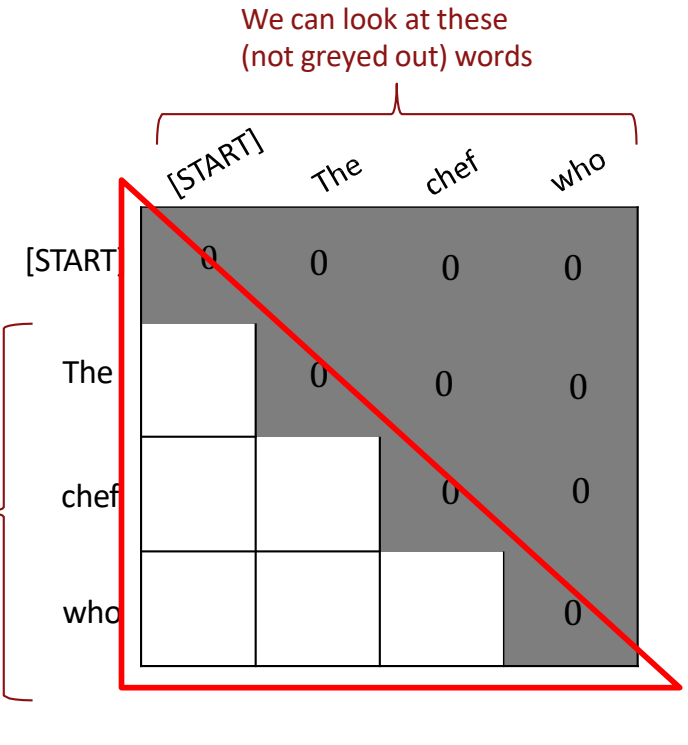
Barriers	Solutions
• Doesn't have an inherent notion of order!	• Add position representations to the inputs
• No nonlinearities for deep learning magic! It's all just weighted averages	• Easy fix: apply the same feedforward network to each self-attention output.

• Need to ensure we don't "look at the future" when predicting a sequence
  • Like in machine translation
  • Or language modeling	

Masking the future in self-attention

- To use self-attention in **decoders**, we need to ensure we can't peek at the future.
- At every timestep, we could change the set of **keys and queries** to include only past words. (Inefficient!)
- To enable parallelization, we **mask out attention** to future words by setting attention scores to 0.

$$e_{ij} = \begin{cases} q_i^\top k_j, j < i \\ 0, j \geq i \end{cases}$$



Barriers and solutions for Self-Attention as a building block

Barriers	Solutions
• Doesn't have an inherent notion of order!	• Add position representations to the inputs
• No nonlinearities for deep learning magic! It's all just weighted averages	• Easy fix: apply the same feedforward network to each self- attention output.

• Need to ensure we don't "look at the future" when predicting a sequence
  • Like in machine translation
  • Or language modeling
	• Mask out the future by artificially setting attention weights to 0!

Necessities for a self-attention building block:

- **Self-attention**:
  - the basis of the method.
- **Position representations**:
  - Specify the sequence order, since self-attention is an unordered function of its inputs.
- **Nonlinearities**:
  - At the output of the self-attention block
  - Frequently implemented as a simple feed-forward network.
- **Masking**:
  - In order to parallelize operations while not looking at the future.
  - Keeps information about the future from "leaking" to the past.

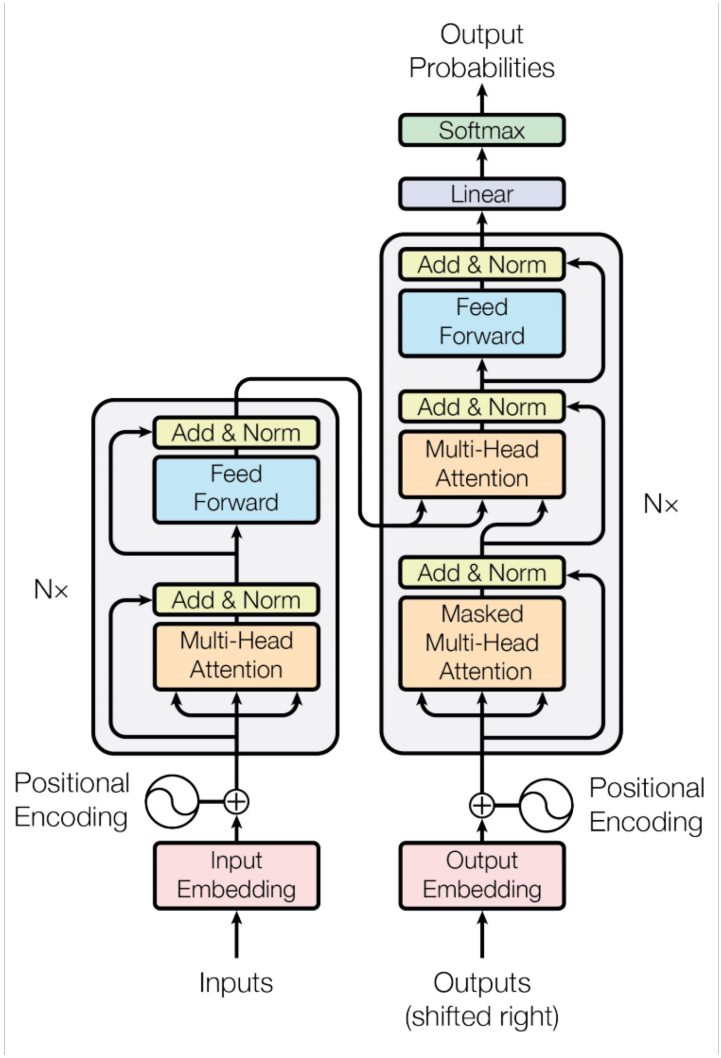
- That's it! But this is not the **Transformer** model we've been hearing about.

Transformer Overview

Attention is all you need. 2017. Aswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, Polosukhin
https://arxiv.org/pdf/1706.03762.pdf

- Non-recurrent sequence-to-sequence encoder-decoder model
- Task: machine translation with parallel corpus
- Predict each translated word
- Final cost/error function: Standard cross-entropy error on top of a softmax classifier

This and related figures from paper ⇑



The Transformer Encoder: Dot-Product Attention

- Inputs: a query q and a set of key-value (k-v) pairs to an output
- Query, keys, values, and output are all vectors

- Output is weighted sum of values, where
- Weight of each value is computed by an inner product of query and corresponding key
- Queries and keys have same dimensionality d_k, value have d_v

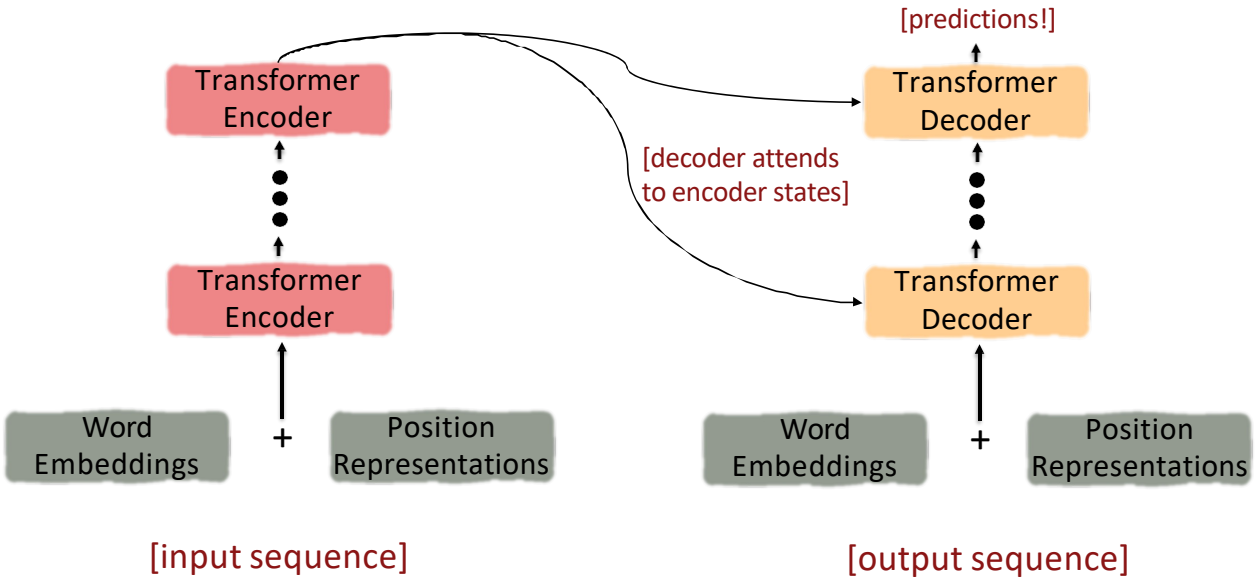
$$A(q, K, V) = \sum_i \frac{e^{q \cdot k_i}}{\sum_j e^{q \cdot k_j}} v_i$$

The Transformer Encoder: Key-Query-Value Attention

- We saw that self-attention is when keys, queries, and values come from the same source. The Transformer does this in a particular way:
  - Let $x_1, \ldots, x_T$ be input vectors to the Transformer encoder; $x_i \in \mathbb{R}^d$
- Then keys, queries, values are:
  - $k_i = Kx_i$, where $K \in \mathbb{R}^{d \times d}$ is the key matrix.
  - $q_i = Qx_i$, where $\mathrm{Q} \in \mathbb{R}^{d \times d}$ is the query matrix.
  - $v_i = Vx_i$, where $\mathrm{V} \in \mathbb{R}^{d \times d}$ is the value matrix.
- These matrices allow *different aspects* of the x vectors to be used/emphasized in each of the three roles.

The Transformer Encoder-Decoder [Vaswani et al., 2017]

Looking back at the whole model, zooming in on an Encoder block:

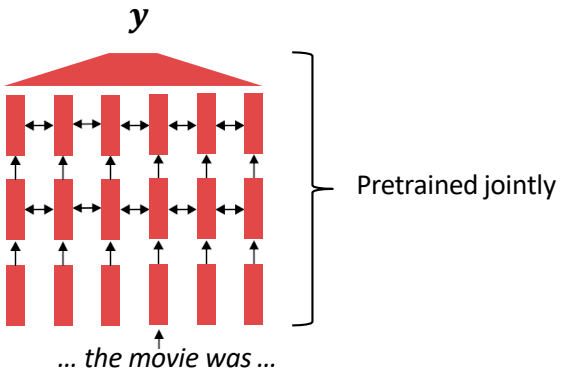


Pretraining models

In modern NLP:

- All (or almost all) parameters in NLP networks are initialized via **pretraining**.
- Pretraining methods hide parts of the input from the model, and train the model to reconstruct those parts.

- This has been exceptionally effective at building strong:
  - **representations of language**
  - **parameter initializations** for strong NLP models.



[This model has learned how to represent entire sentences through pretraining]

Adapted from John Hewitt

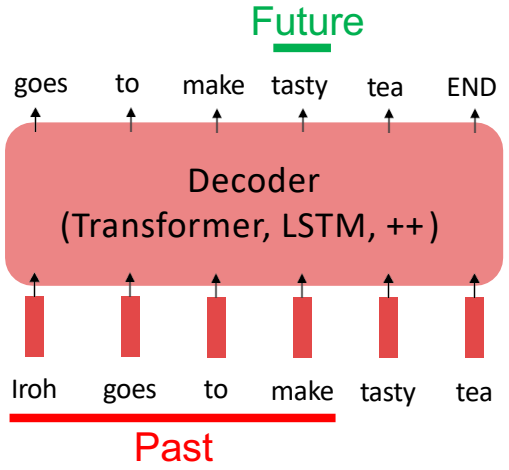
Pretraining through language modeling [Dai and Le, 2015]

Recall the **language modeling** task:

- Model $p_\theta(w_t \mid w_{1:t-1})$, the probability distribution over words given their past contexts.
- There's lots of data for this! (In English.)

Pretraining through language modeling:

- Train a neural network to perform language modeling on a large amount of text.
- Save the network parameters.



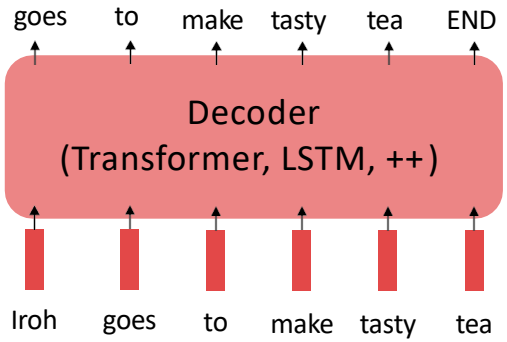
See on our

The Pretraining / Finetuning Paradigm

Pretraining can improve NLP applications by serving as parameter initialization.

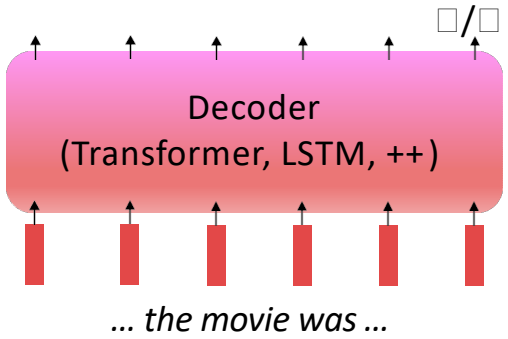
Step 1: Pretrain (on language modeling)

Lots of text; learn general things!



Step 2: Finetune (on your task)

Not many labels; adapt to the task!



Generative Pretrained Transformer (GPT) [Radford et al., 2018]

2018's GPT was a big success in pretraining a decoder!

- Transformer decoder with 12 layers.
- 768-dimensional hidden states, 3072-dimensional feed-forward hidden layers.
- Byte-pair encoding with 40,000 merges
- Trained on **BooksCorpus**: over 7000 unique books.
  - Contains long spans of contiguous text, for learning long-distance dependencies.
- The acronym "GPT" never showed up in the original paper; it could stand for "Generative PreTraining" or "Generative Pretrained Transformer"

[Devlin et al., 2018]

Generative Pretrained Transformer (GPT) [Radford et al., 2018]

How do we format inputs to our decoder for **finetuning tasks?**

Natural Language Inference: Label pairs of sentences as *entailing*/*contradictory*/*neutral*

Premise: *The man is in the doorway*
Hypothesis: *The person is near the door* } **entailment**

Radford et al., 2018 evaluate on natural language inference.
Here's roughly how the input was formatted, as a sequence of tokens for the decoder.

[START] *The man is in the doorway* **[DELIM]** *The person is near the door* **[EXTRACT]**

The linear classifier is applied to the representation of the **[EXTRACT] token**.

GPT-3, in-context learning, very large models

So far, we've interacted with pretrained models in two ways:

- Sample from the distributions they define (maybe providing a prompt)
- Fine-tune them on a task we care about, and take their predictions.

Very large language models seem to perform some kind of learning **without gradient steps** simply from examples you provide within their contexts.

GPT-3 is the canonical example of this.

GPT-3 has 175 billion parameters.

Adapted from John Hewitt

GPT-3, in-context learning, very large models

Very large language models seem to perform some kind of learning **without gradient steps** simply from examples you provide within their contexts.

The in-context examples seem to specify the task to be performed, and the conditional distribution mocks performing the task to a certain extent.

Input (prefix within a single Transformer decoder context):

“ thanks -> merci
hello -> bonjour
mint -> menthe
otter -> ”

Output (conditional generations):

loutre…”

John Hewitt

GPT-3: Prompt Engineering

Translate English to French

```
sea otter => loutre de mer

peppermint => menthe poivrée

plush girafe => girafe peluche

cheese =>
```

Language Models are Few-Shot Learners

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, Dario Amodei

https://arxiv.org/abs/2005.14165

Prompt Engineering

Prompt engineering

12 languages

Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

Prompt engineering is a concept in artificial intelligence (AI), particularly natural language processing (NLP). In prompt engineering, the description of the task that the AI is supposed to accomplish is embedded in the input, e.g., as a question, instead of it being implicitly given. Prompt engineering typically works by converting one or more tasks to a prompt-based dataset and training a language model with what has been called "prompt-based learning" or just "prompt learning".[1][2]

History [edit]

The GPT-2 and GPT-3 language models[3] were important steps in prompt engineering. In 2021, multitask[*jargon*] prompt engineering using multiple NLP datasets showed good performance on new tasks.[4] In a method called chain-of-thought (CoT) prompting, few-shot examples of a task are given to the language model which improves its ability to reason.[5] CoT prompting can also be a zero-shot learning task by prepending text to the prompt that encourages a chain of thought (e.g. "Let's think step by step"), which may also improve the performance of a language model in multi-step reasoning problems.[6] The broad accessibility of these tools were driven by the publication of several open-source notebooks and community-led projects for image synthesis.[7]

A description for handling prompts reported that over 2,000 public prompts for around 170 datasets were available in February 2022.[8]

How would you come with a solution for this problem?

The kid is throwing rocks at the window

The <subject>kid</subject> is throwing <object>rocks</object> at the <destination>window</destination>

Prompt Engineering

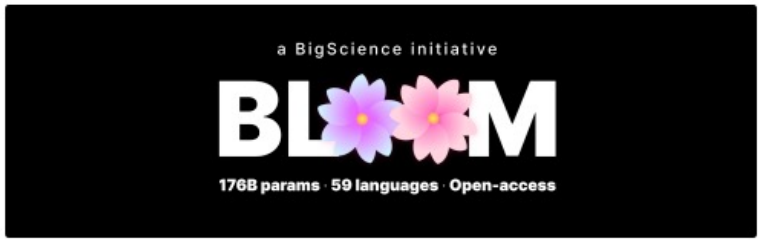
Input: The cat is throwing the ball into the ground
Output: The <subject>cat</subject> is throwing the <object>ball</object> into the <destination>ground</ground>

Input: The snake is being attacked by the wolf
Output: The <object>snake</object> is being attacked by the <actor>wolf</actor>

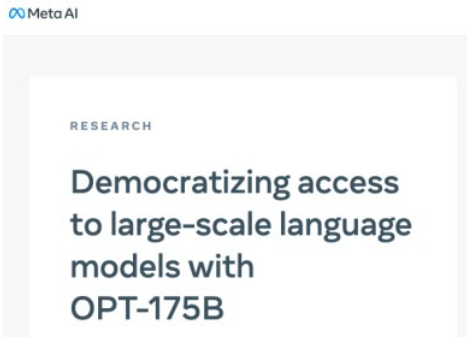
Input: The kid is throwing rocks at the window
Output:

Prompt Engineering

- Any Large Language Model (LLM) such as GPT-3 can be turned into a general purpose problem solver in this way.
- Obviously, it is not going to work well for every use case.
- Other Large Language Models trained at the scale of GPT-3 that are actually publicly available.
- BLOOM-176B and OPT-175B:



https://huggingface.co/bigscience/bloom



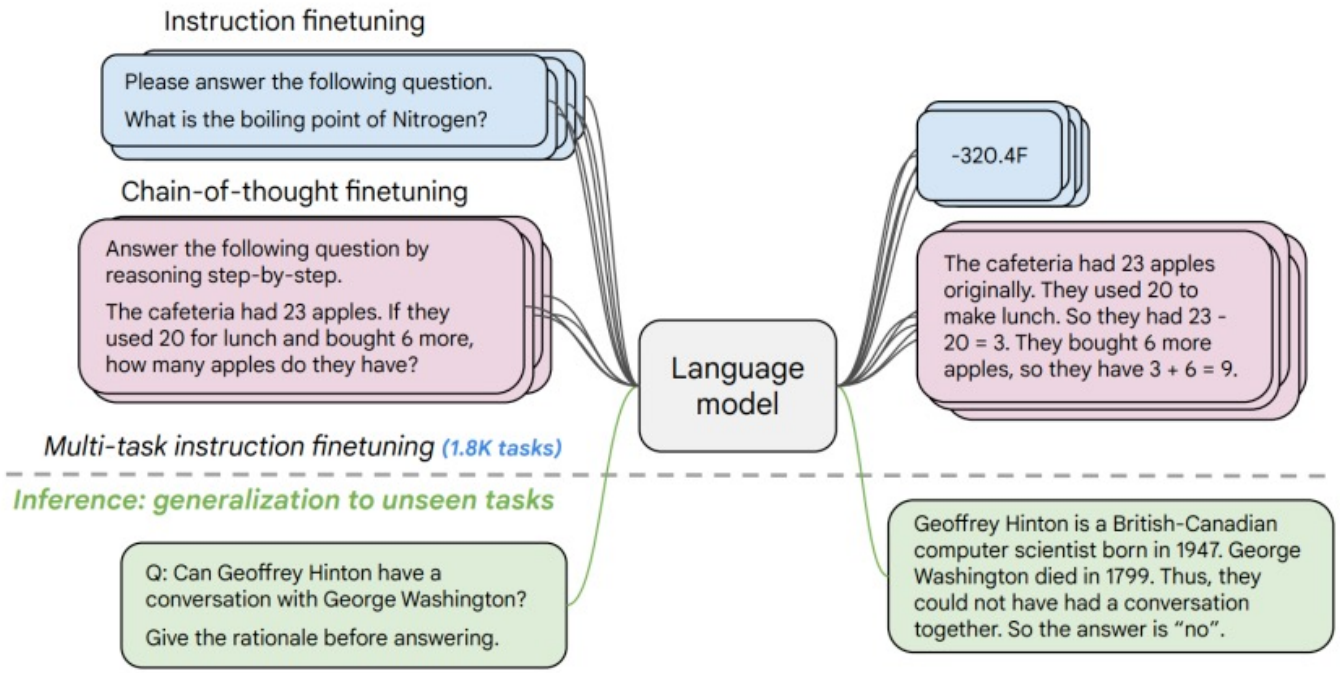
https://huggingface.co/docs/transformers/model_doc/opt

However these are still limited

- Predicting the next word can lead to intelligent behavior such as the one exemplified earlier however this still limited

- What makes some of the new LLMs special? ChatGPT (GPT-3.5, 3.5 Turbo, 4, 4-turbo), FLAN-T5, OPT-IML

Instruction Tuning (e.g. FLAN-T5 by Google)



See on our

https://arxiv.org/pdf/2210.11416.pdf

FLAN-T5

Model input (Disambiguation QA)

Q: In the following sentences, explain the antecedent of the pronoun (which thing the pronoun refers to), or state that it is ambiguous.

Sentence: The reporter and the chef will discuss their favorite dishes.

Options:
(A) They will discuss the reporter's favorite dishes
(B) They will discuss the chef's favorite dishes
(C) Ambiguous

A: Let's think step by step.

Before instruction finetuning

The reporter and the chef will discuss their favorite dishes.
The reporter and the chef will discuss the reporter's favorite dishes.
The reporter and the chef will discuss the chef's favorite dishes.
The reporter and the chef will discuss the reporter's and the chef's favorite dishes.

✖ **(doesn't answer question)**

FLAN-T5

Model input (Disambiguation QA)

Q: In the following sentences, explain the antecedent of the pronoun (which thing the pronoun refers to), or state that it is ambiguous.

Sentence: The reporter and the chef will discuss their favorite dishes.

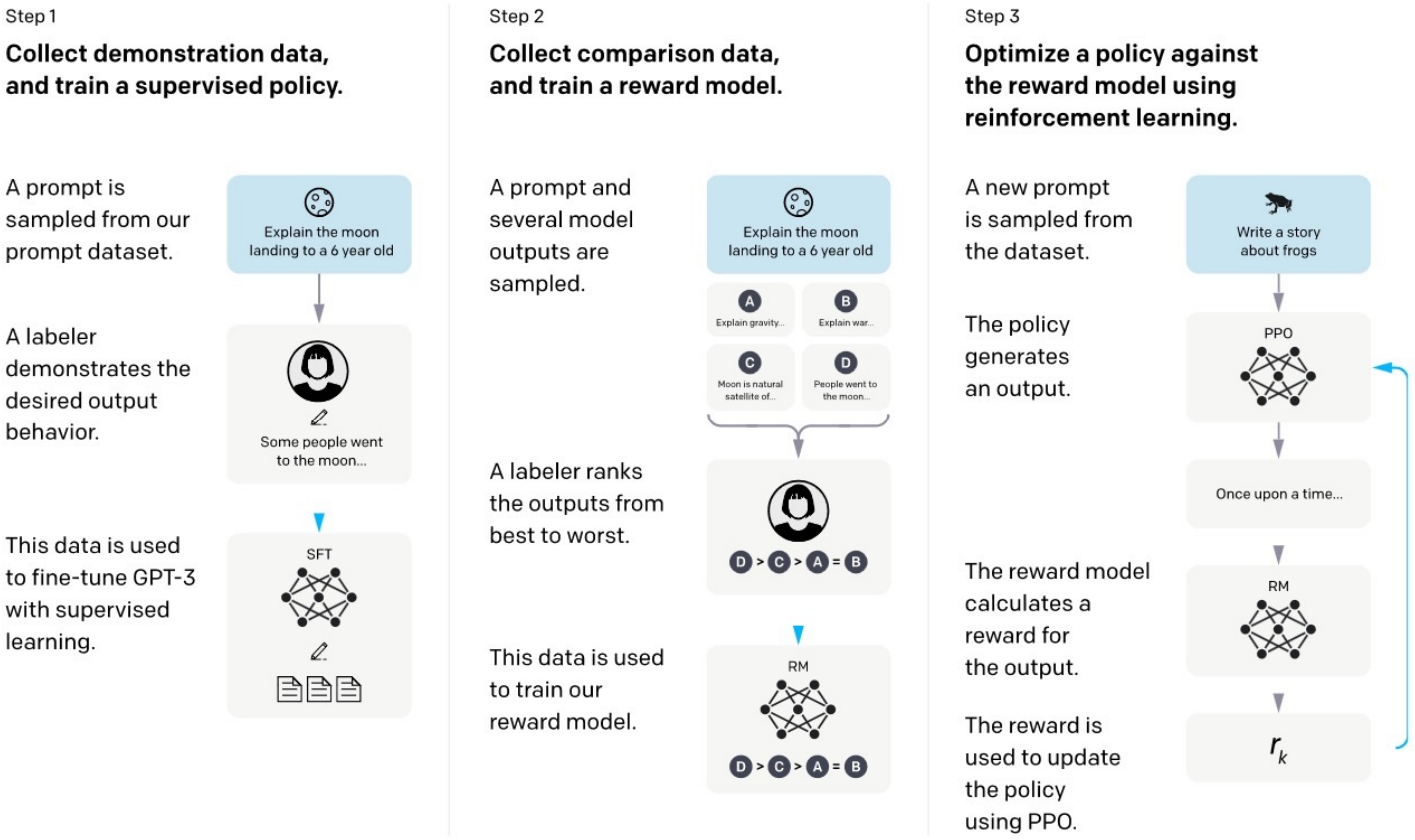
Options:
(A) They will discuss the reporter's favorite dishes
(B) They will discuss the chef's favorite dishes
(C) Ambiguous

A: Let's think step by step.

After instruction finetuning

The reporter and the chef will discuss their favorite dishes does not indicate whose favorite dishes they will discuss. So, the answer is (C). ✅

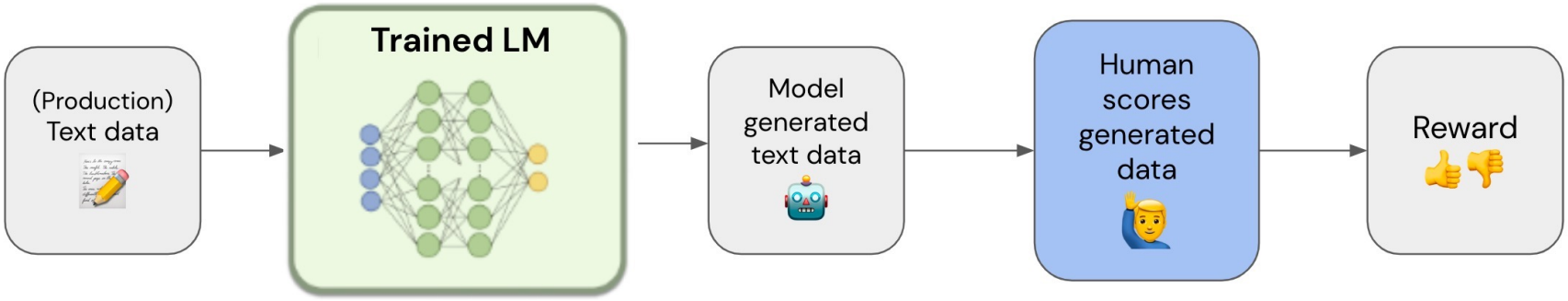
InstructGPT (ChatGPT)



https://arxiv.org/abs/2203.02155

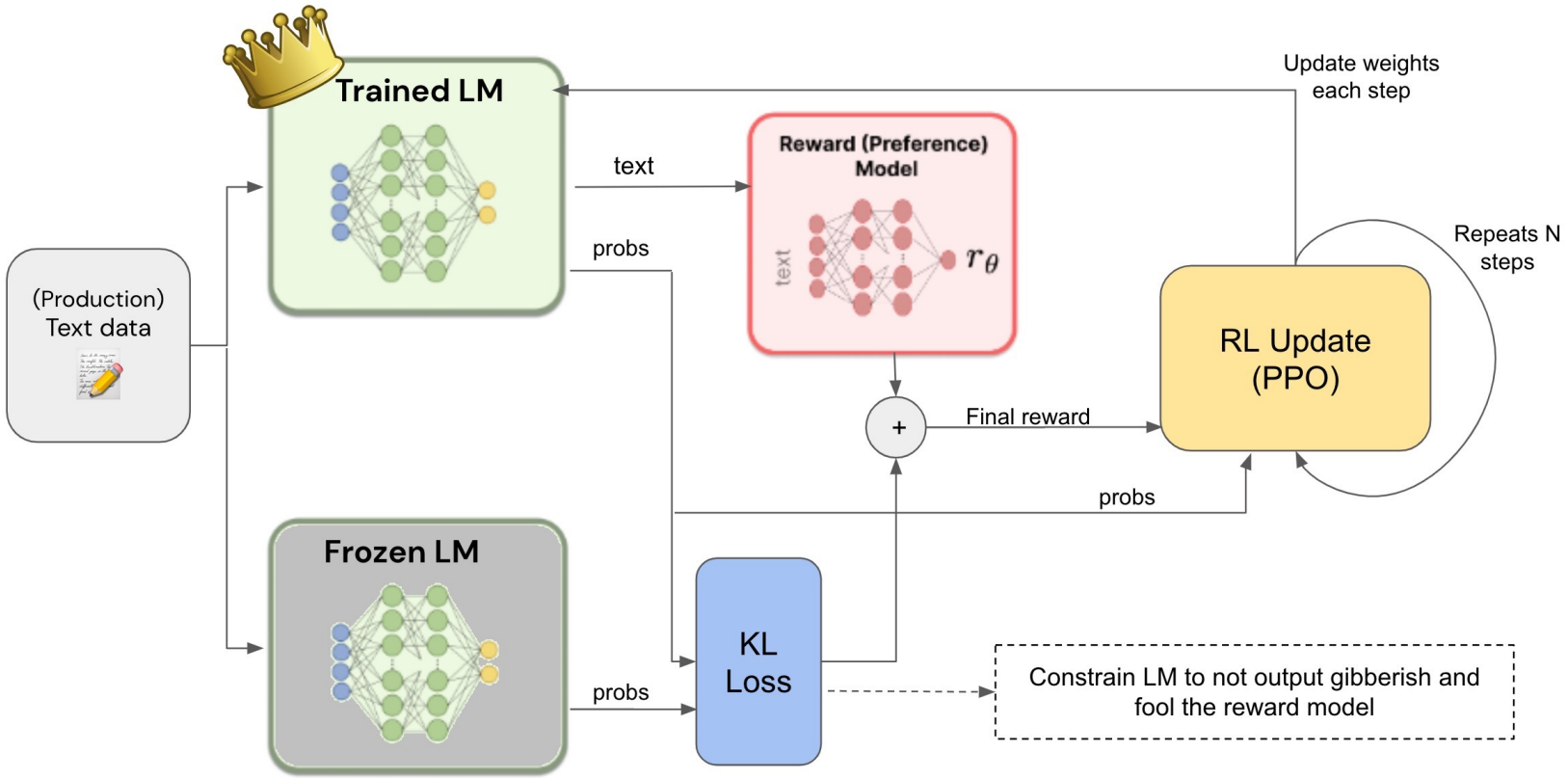
Vicente Ordoñez

Step by step: Train a reward model that learns from Human Ratings (e.g. from 1 to 5)



https://gist.github.com/JoaoLages/c6f2dfd13d2484aa8bb0b2d567fbf093

Step by step: Train the LM to generate text that get high reward but still produces stuff that makes sense



https://gist.github.com/JoaoLages/c6f2dfd13d2484aa8bb0b2d567fbf093

Vicente Ordoñez

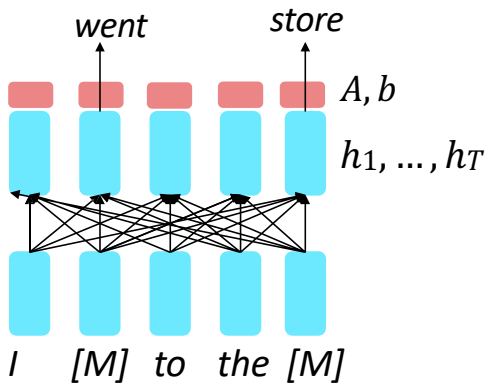
Pretraining encoders: What pretraining objective to use?

So far, we've looked at language model pretraining. But **encoders get bidirectional context,** so we can't do language modeling!

Idea: replace some fraction of words in the input with a special [MASK] token; predict these words.

$$h_1, \dots, h_T = \text{Encoder}(w_1, \dots, w_T)$$

$$y_i \sim Aw_i + b$$



[Devlin et al., 2018]

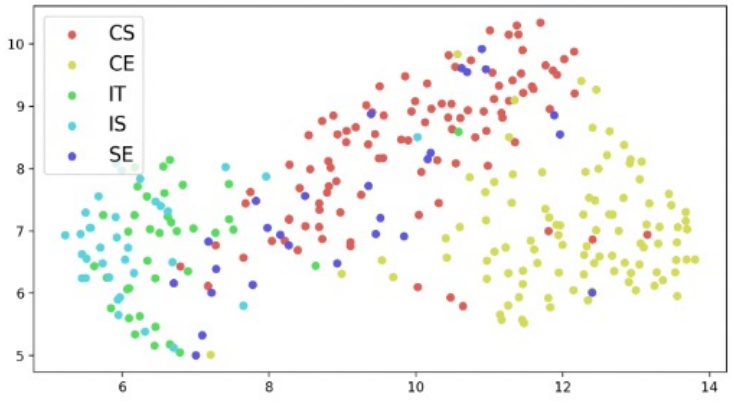
Only add loss terms from words that are "masked out." If x' is the masked version of x, we're learning $p_\theta(x|x')$. Called **Masked LM**.

Example: BERT: Bidirectional Encoder Representations from Tranformers

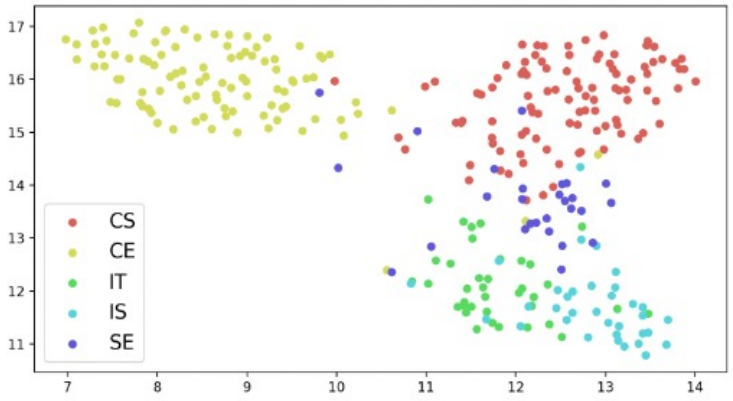
Adapted from John Hewitt

Case Study: Improving Embeddings Representations for Comparing Higher Education Curricula

- Umap (MacInnes et al, JOSS 2018) visualizations for Bert and our approach.
- Our approach separates computing programs more clearly.



(a) *Bert*



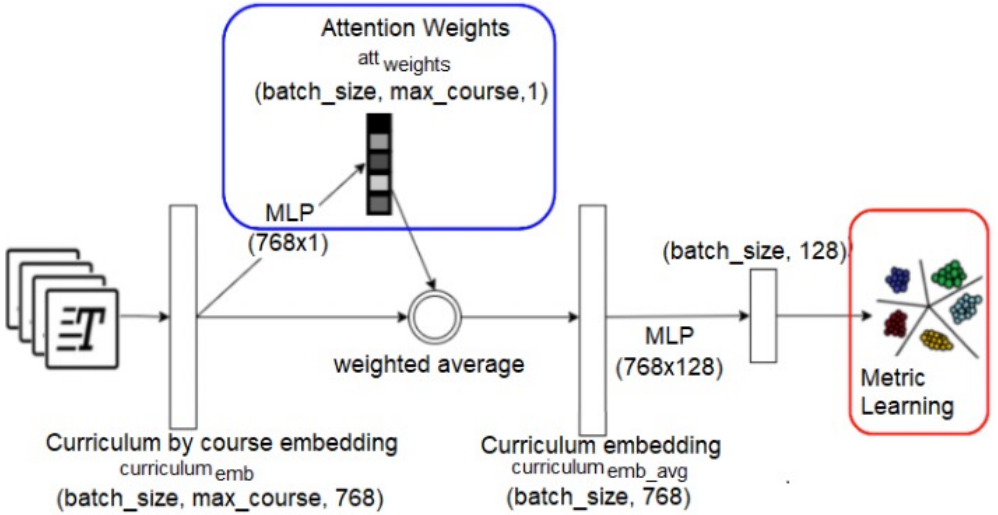
(b) $Bert_{met+att}$

Murrugarra-Llerena et al, EMNLP 2022

Case Study: Improving Embeddings Representations for Comparing Higher Education Curricula

- Course-Based attention: Identifies the most and the least important courses following the intuition of core and elective courses.

- Metric Learning: Learns boundaries to form well-defined groups.



Murrugarra-Llerena et al, EMNLP 2022

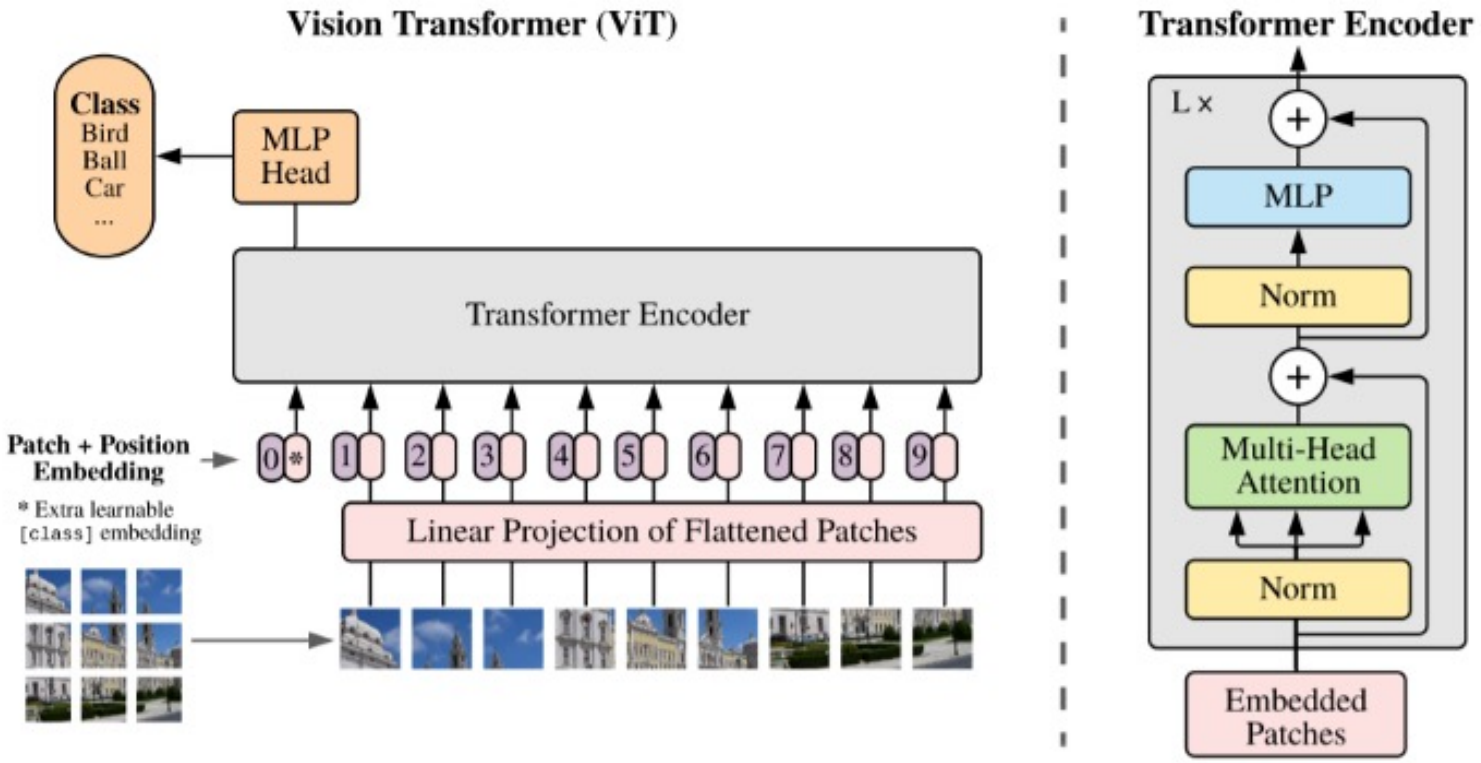
Capturing meaning via context: What kinds of things does pretraining learn?

There's increasing evidence that pretrained models learn a wide variety of things about the statistical properties of language:

- *Stanford University is located in_____, California.* [Trivia]
- *I put___fork down on the table.* [syntax]
- *The woman walked across the street, checking for traffic over_____ shoulder.* [coreference]
- *I went to the ocean to see the fish, turtles, seals, and_____.* [lexical semantics/topic]
- *Overall, the value I got from the two hours watching it was the sum total of the popcorn and the drink. The movie was___.* [sentiment]
- Iroh went into the kitchen to make some tea. Standing next to Iroh, Zuko pondered his destiny. Zuko left the_____. [some reasoning – this is harder]
- I was thinking about the sequence that goes 1, 1, 2, 3, 5, 8, 13, 21,__ [some basic arithmetic; they don't learn the Fibonnaci sequence]
- Models also learn – and can exacerbate racism, sexism, all manner of bad biases.

Adapted from John Hewitt

Transformers in vision



Dosovitskiy, ICLR 2021, https://github.com/google-research/vision_transformer

https://www.youtube.com/watch?v=TrdevFK_am4

Cross-modal transformers

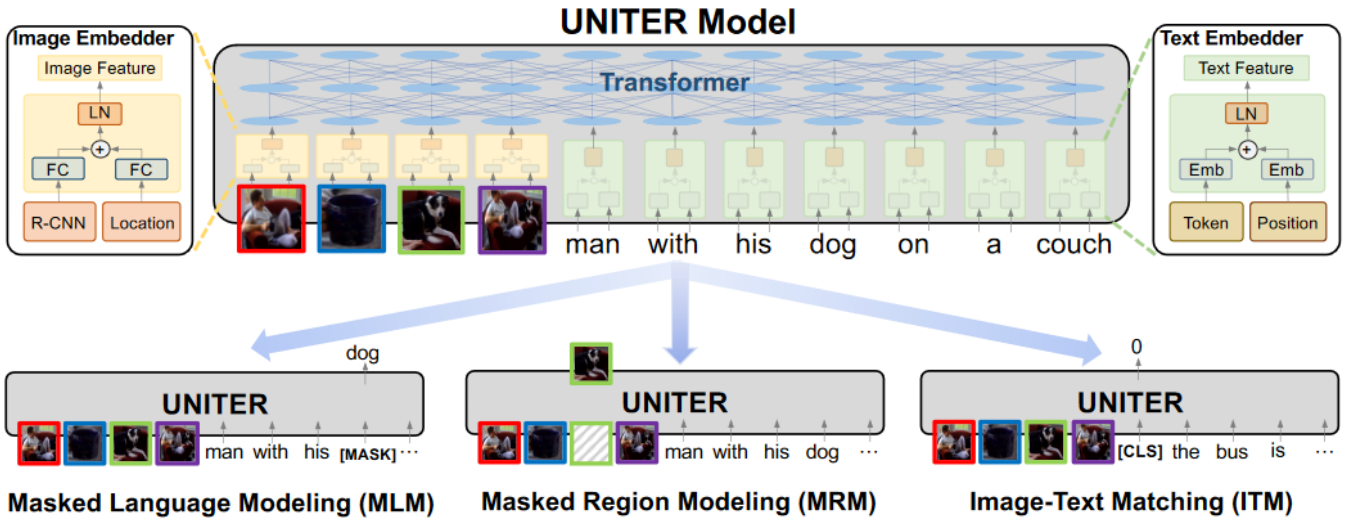
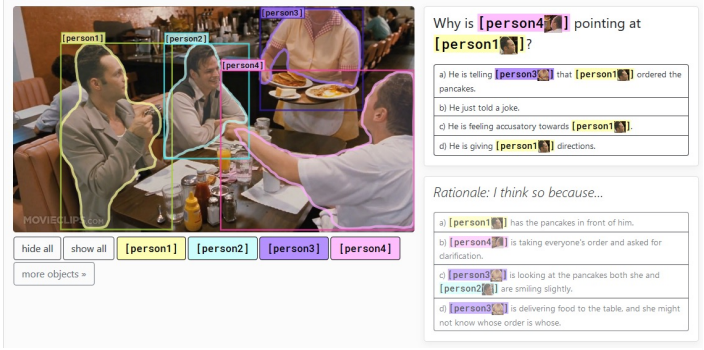


Figure 1: Overview of the proposed UNITER model (best viewed in color), consisting of an Image Embedder, a Text Embedder and a multi-layer self-attention Transformer, learned through three pre-training tasks.

Chen et al., "UNITER: Learning UNiversal Image-TExt Representations", arxiv 2019

Visual Commonsense Reasoning leaderboard



https://visualcommonsense.com/leaderboard/

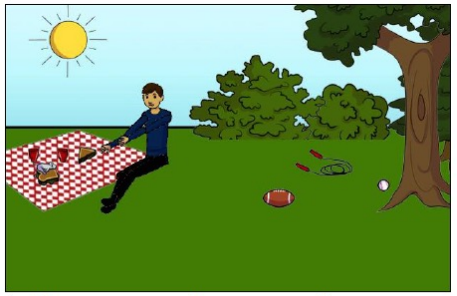
Rank	Model	Q->A	QA->R	Q->AR
	Human Performance *University of Washington* (Zellers et al. '18)	91.0	93.0	85.0
new September 30, 2019	UNITER-large (ensemble) *MS D365 AI* https://arxiv.org/abs/1909.11740	**79.8**	**83.4**	**66.8**
2 September 23, 2019	UNITER-large (single model) *MS D365 AI* https://arxiv.org/abs/1909.11740	77.3	80.8	62.8
3 August 9,2019	ViLBERT (ensemble of 10 models) *Georgia Tech & Facebook AI Research* https://arxiv.org/abs/1908.02265	76.4	78.0	59.8
4 September 23,2019	VL-BERT (single model) *MSRA & USTC* https://arxiv.org/abs/1908.08530	75.8	78.4	59.7
5 August 9,2019	ViLBERT (ensemble of 5 models) *Georgia Tech & Facebook AI Research* https://arxiv.org/abs/1908.02265	75.7	77.5	58.8

Visual Question Answering (VQA)

Task: Given an image and a natural language open-ended question, generate a natural language answer.

What color are her eyes?
What is the mustache made of?

How many slices of pizza are there?
Is this a vegetarian pizza?

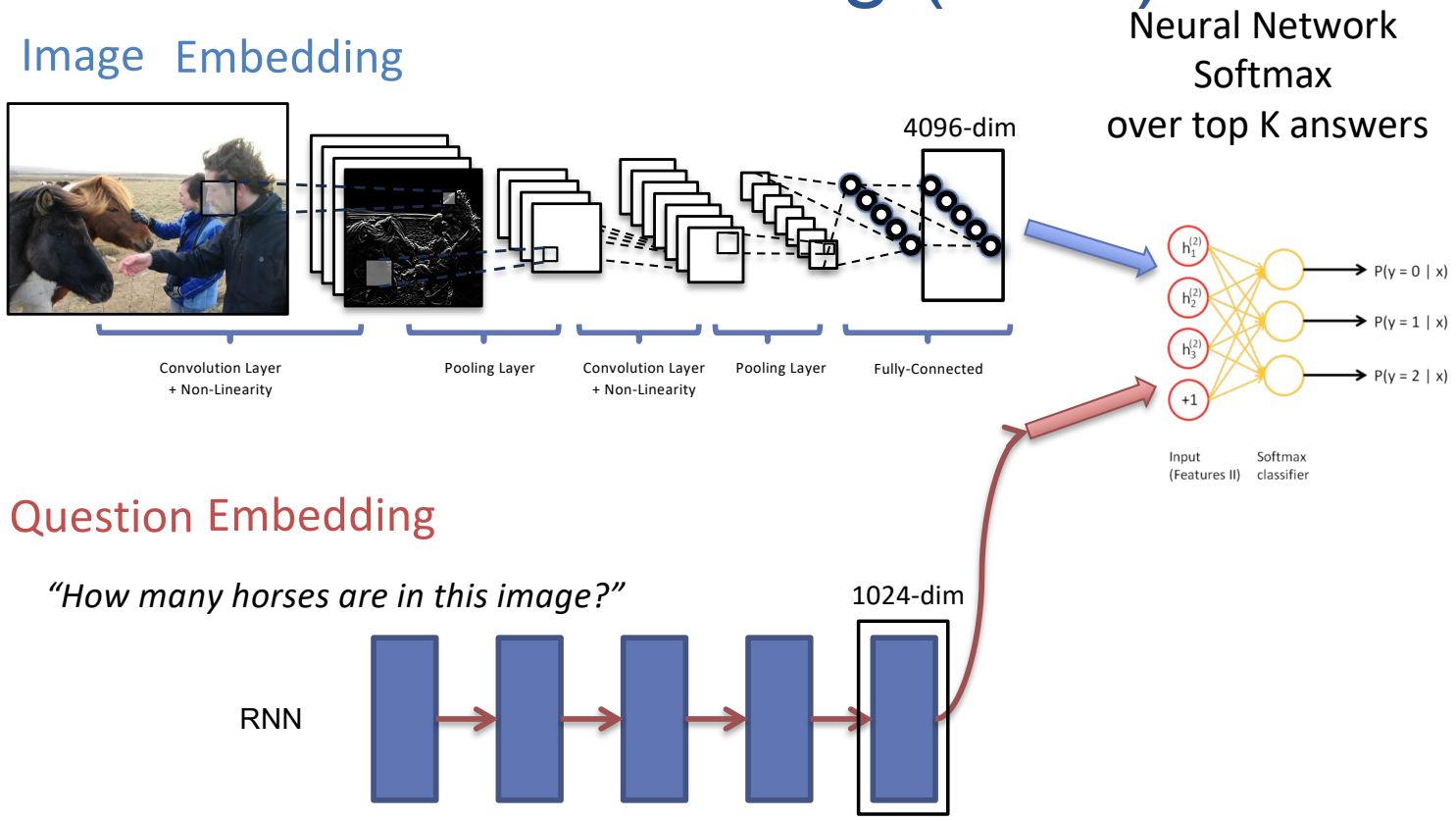


Is this person expecting company?
What is just under the tree?

Does it appear to be rainy?
Does this person have 20/20 vision?

Agrawal et al., "VQA: Visual Question Answering", ICCV 2015

Visual Question Answering (VQA)



Agrawal et al., "VQA: Visual Question Answering", ICCV 2015

Plan for this lecture

- Language and vision
  - Application: Image and video captioning
  - Tool: Recurrent neural networks
  - Tool: Transformers
  - Application: Visual question answering
- Motion and video
  - Video classification
  - Measuring motion
  - Tracking objects

Video Classification

Input video:
T x 3 x H x W

Swimming
Running
Jumping
Eating
Standing

Running video is in the public domain

Slide credit: Justin Johnson

Problem: Videos are big!

Input video:
T x 3 x H x W

Videos are ~30 frames per second (fps)

Size of uncompressed video (3 bytes per pixel):

SD (640 x 480): **~1.5 GB per minute**
HD (1920 x 1080): **~10 GB per minute**

Solution: Train on short **clips:** low fps and low spatial resolution
e.g. T = 16, H=W=112
(3.2 seconds at 5 fps, 588 KB)

Slide credit: Justin Johnson

Training on Clips

Raw video: Long, high FPS

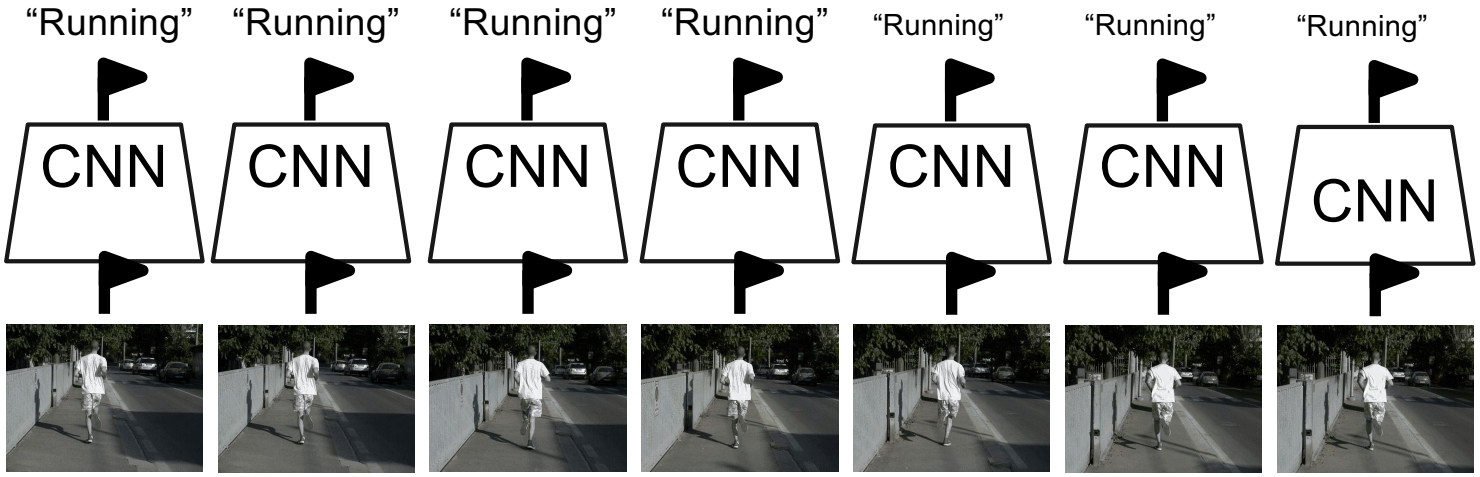
Training: Train model to classify short **clips** with low FPS

Testing: Run model on different clips, average predictions

Slide credit: Justin Johnson

Video Classification: Single-Frame CNN

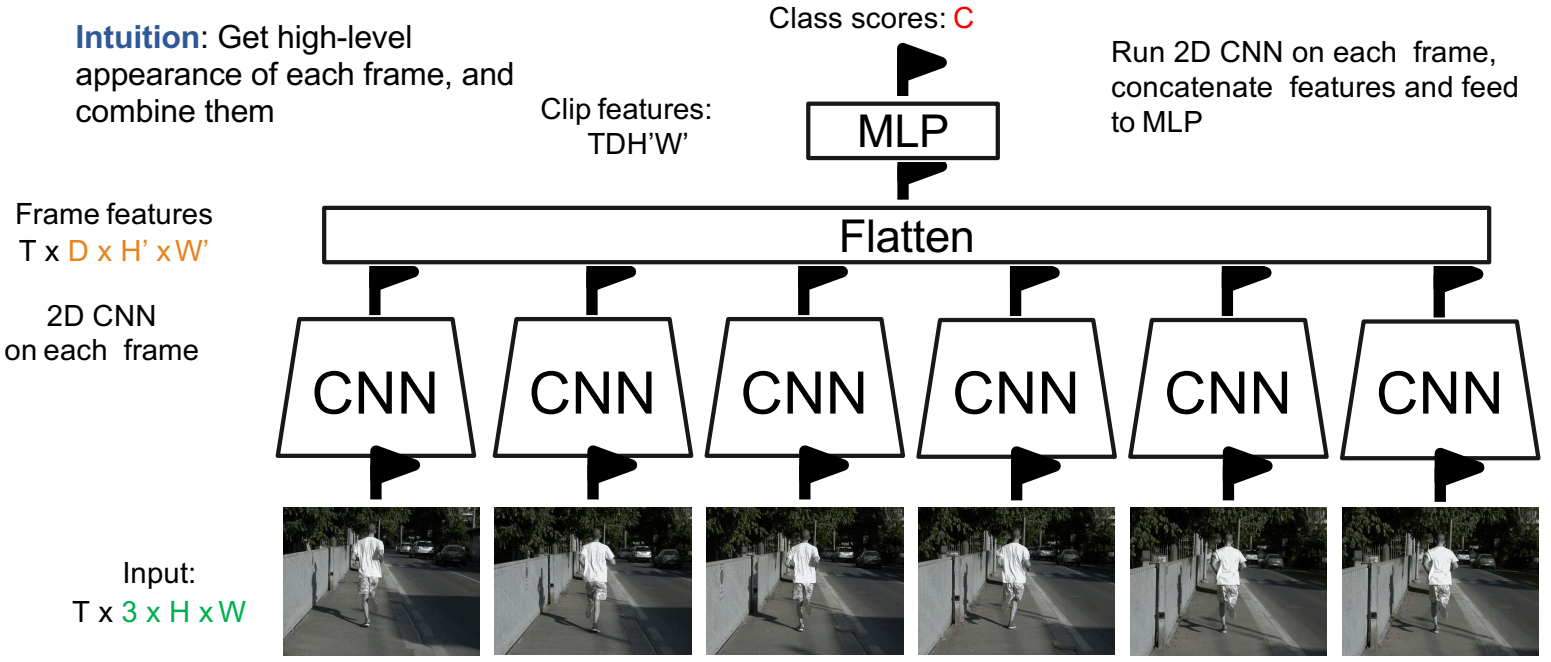
Simple idea: train normal 2D CNN to classify video frames independently! (Average predicted probs at test-time)
Often a **very** strong baseline for video classification



Slide credit: Justin Johnson

Video Classification: Late Fusion (with FC layers)

Intuition: Get high-level appearance of each frame, and combine them



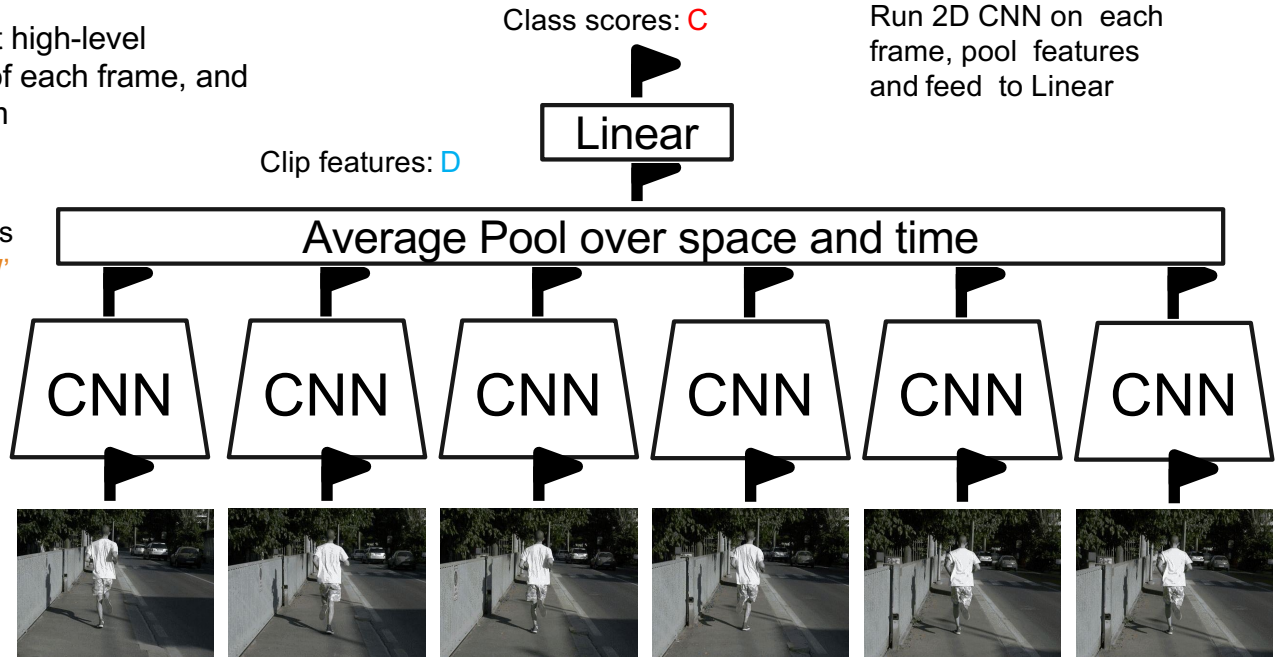
Run 2D CNN on each frame, concatenate features and feed to MLP

Karpathy et al, "Large-scale Video Classification with Convolutional Neural Networks", CVPR 2014

Slide credit: Justin Johnson

Video Classification: Late Fusion (with pooling)

Intuition: Get high-level appearance of each frame, and combine them

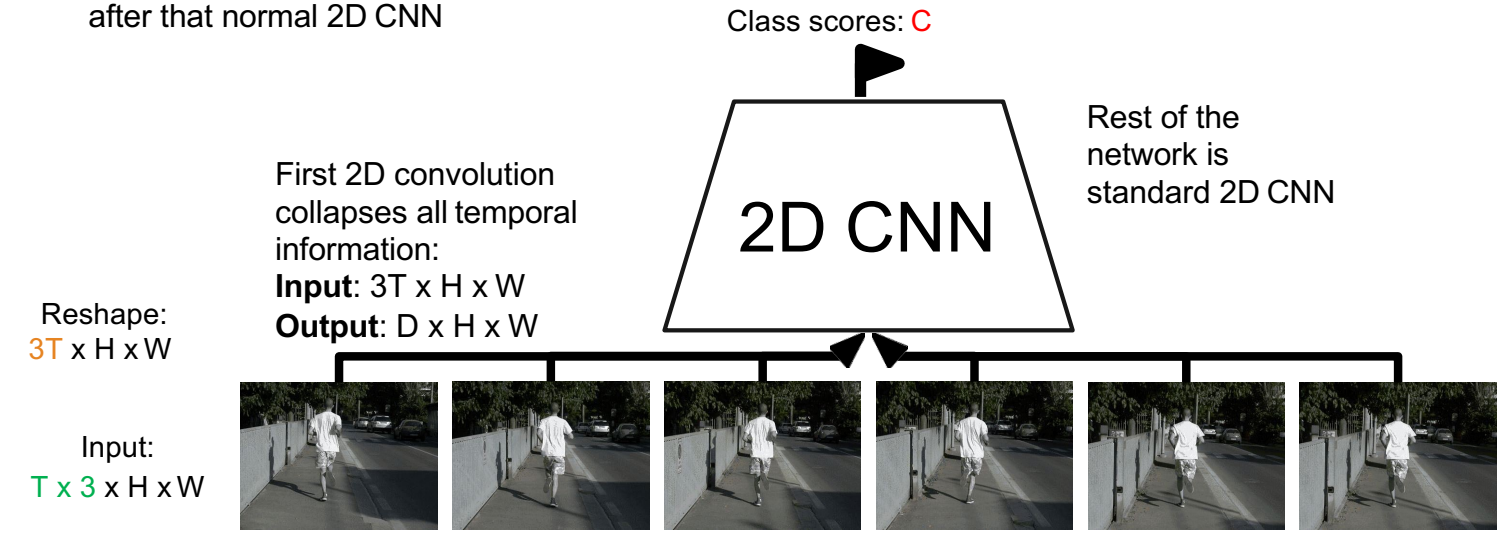


Run 2D CNN on each frame, pool features and feed to Linear

Slide credit: Justin Johnson

Video Classification: Early Fusion

Intuition: Compare frames with very first conv layer, after that normal 2D CNN



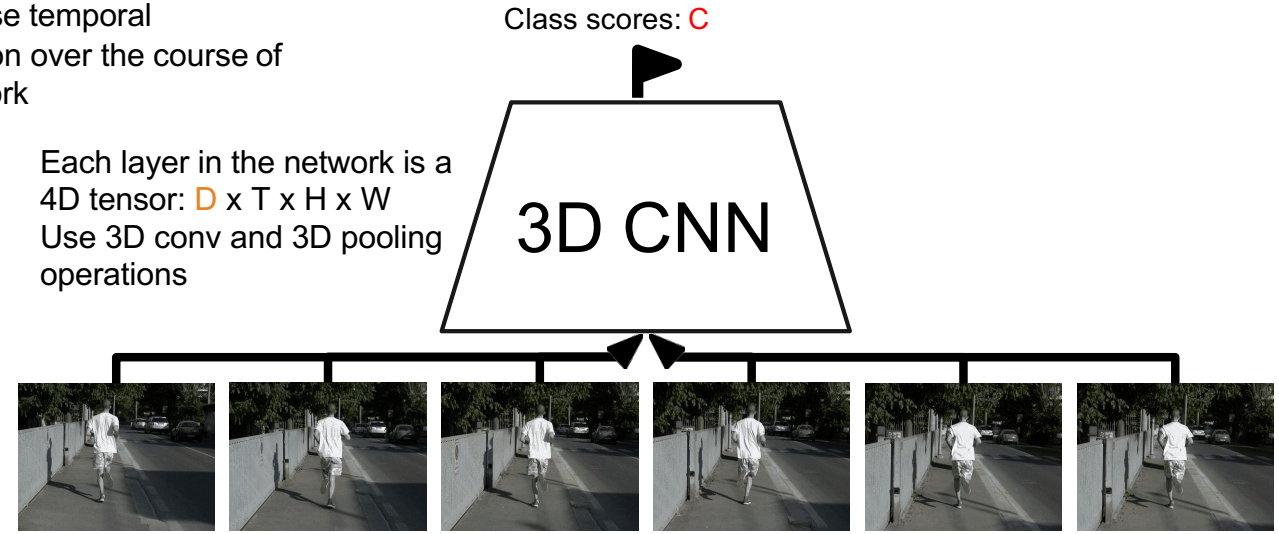
Karpathy et al, "Large-scale Video Classification with Convolutional Neural Networks", CVPR 2014

Slide credit: Justin Johnson

Video Classification: 3D CNN

Intuition: Use 3D versions of convolution and pooling to slowly fuse temporal information over the course of the network

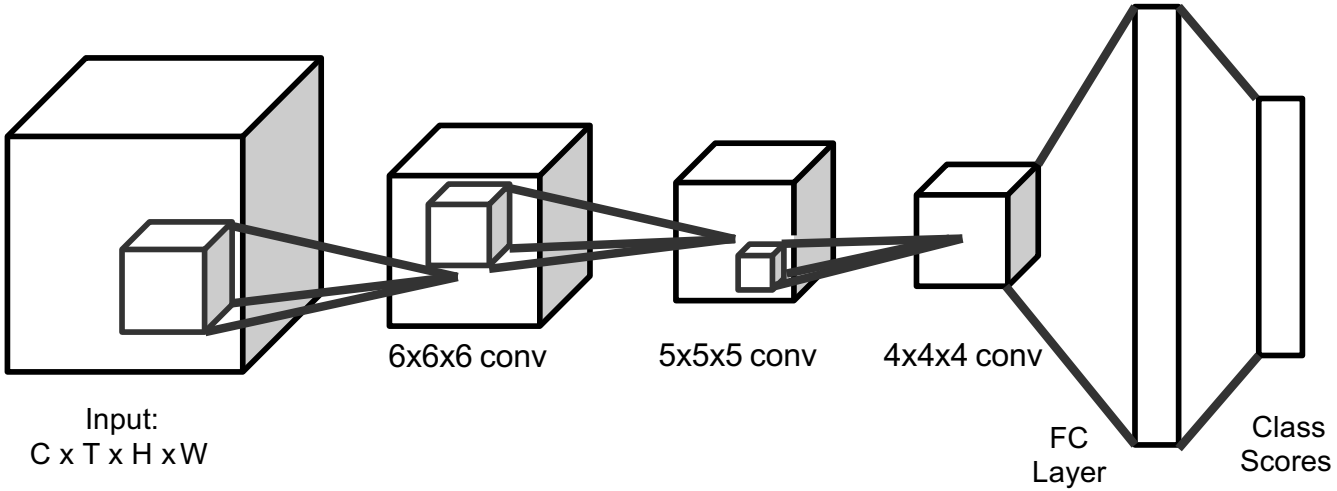
Each layer in the network is a 4D tensor: D x T x H x W
Use 3D conv and 3D pooling operations



Ji et al, "3D Convolutional Neural Networks for Human Action Recognition", TPAMI 2010 ; Karpathy et al, "Large-scale Video Classification with Convolutional Neural Networks", CVPR 2014

Slide credit: Justin Johnson

3D Convolution



Slide credit: Fei-Fei Li, Yunzhu Li, Ruohan Gao

C3D: The VGG of 3D CNNs

3D CNN that uses all 3x3x3 conv and 2x2x2 pooling
(except Pool1 which is 1x2x2)

Released model pretrained on Sports-1M:
Many people used this as a video feature extractor

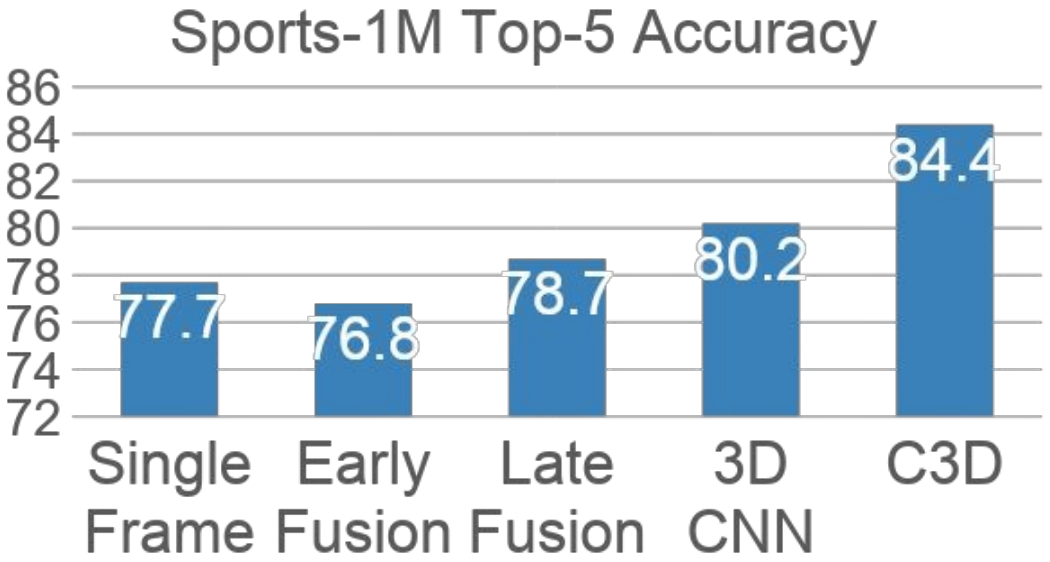
See on our

Layer	Size
Input	3 x 16 x 112 x 112
Conv1 (3x3x3)	64 x 16 x 112 x 112
Pool1 (1x2x2)	64 x 16 x 56 x 56
Conv2 (3x3x3)	128 x 16 x 56 x 56
Pool2 (2x2x2)	128 x 8 x 28 x 28
Conv3a (3x3x3)	256 x 8 x 28 x 28
Conv3b (3x3x3)	256 x 8 x 28 x 28
Pool3 (2x2x2)	256 x 4 x 14 x 14
Conv4a (3x3x3)	512 x 4 x 14 x 14
Conv4b (3x3x3)	512 x 4 x 14 x 14
Pool4 (2x2x2)	512 x 2 x 7 x 7
Conv5a (3x3x3)	512 x 2 x 7 x 7
Conv5b (3x3x3)	512 x 2 x 7 x 7
Pool5	512 x 1 x 3 x 3
FC6	4096
FC7	4096
FC8	C

Slide credit: Fei-Fei Li, Yunzhu Li, Ruohan Gao

Tran et al, "Learning Spatiotemporal Features with 3D Convolutional Networks", ICCV 2015

Early Fusion vs Late Fusion vs 3D CNN



Karpathy et al, "Large-scale Video Classification with Convolutional Neural Networks", CVPR 2014
Tran et al, "Learning Spatiotemporal Features with 3D Convolutional Networks", ICCV 2015

Slide credit: Justin Johnson

Motion: Why is it useful?

Derek Hoiem

Motion: Why is it useful?

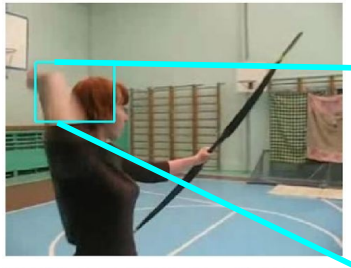
- Even “impoverished” motion data can evoke a strong percept

G. Johansson, “Visual Perception of Biological Motion and a Model For Its Analysis", *Perception and Psychophysics 14, 201-211, 1973.*

Derek Hoiem

Measuring Motion: Optical Flow

Image at frame t



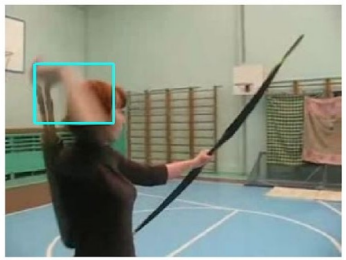
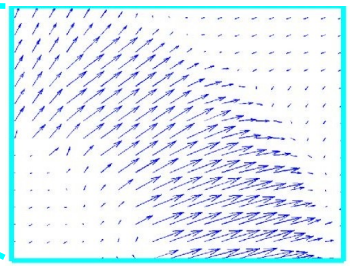


Image at frame t+1

Optical flow gives a

displacement field F between images I_t and I_{t+1}



Tells where each pixel will move in the next frame:

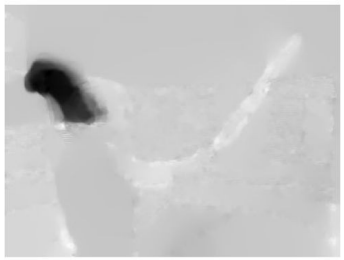
$F(x, y) = (dx, dy)$

$I_{t+1}(x+dx, y+dy) = I_t(x, y)$

Optical Flow highlights

local motion

Horizontal flow dx

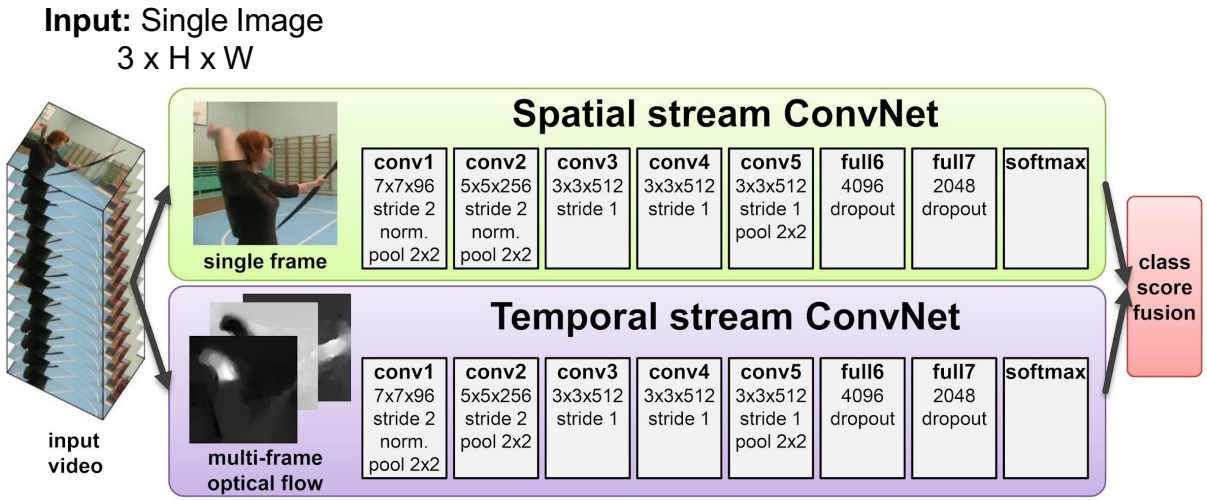


Vertical Flow dy

Simonyan and Zisserman, "Two-stream convolutional networks for action recognition in videos", NeurIPS 2014

Slide credit: Justin Johnson

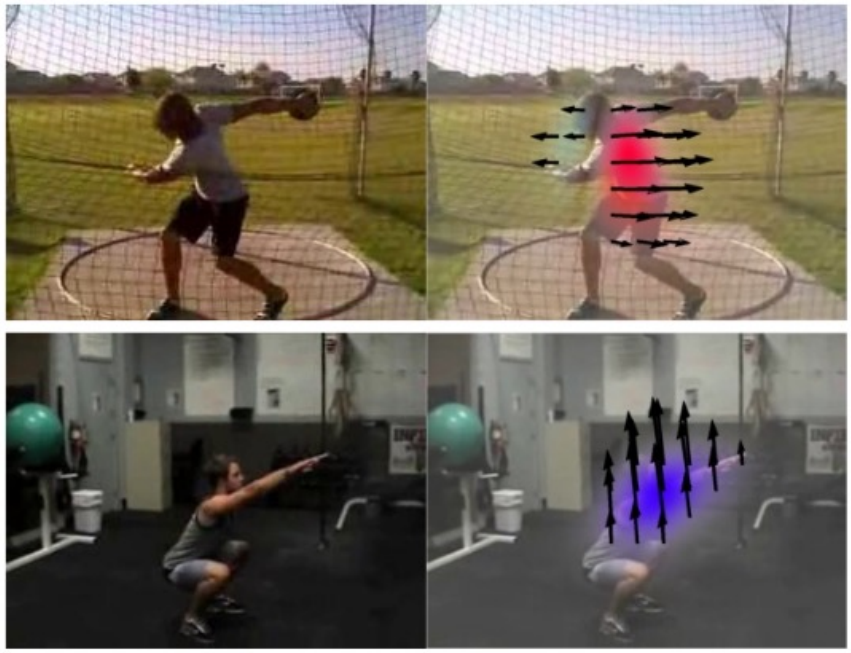
Separating Motion and Appearance: Two-Stream Networks



Simonyan and Zisserman, "Two-stream convolutional networks for action recognition in videos", NeurIPS 2014

Slide credit: Justin Johnson

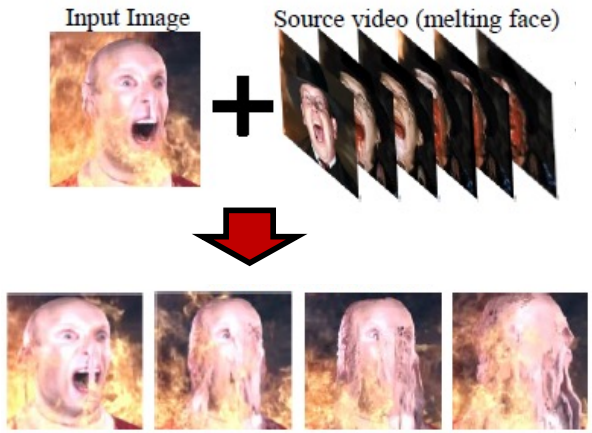
Modeling Motion: Optical Flow



(a) Input Image (b) Prediction

Walker et al., “Dense Optical Flow Prediction from a Static Scene”, ICCV 2015

Transferring Motion

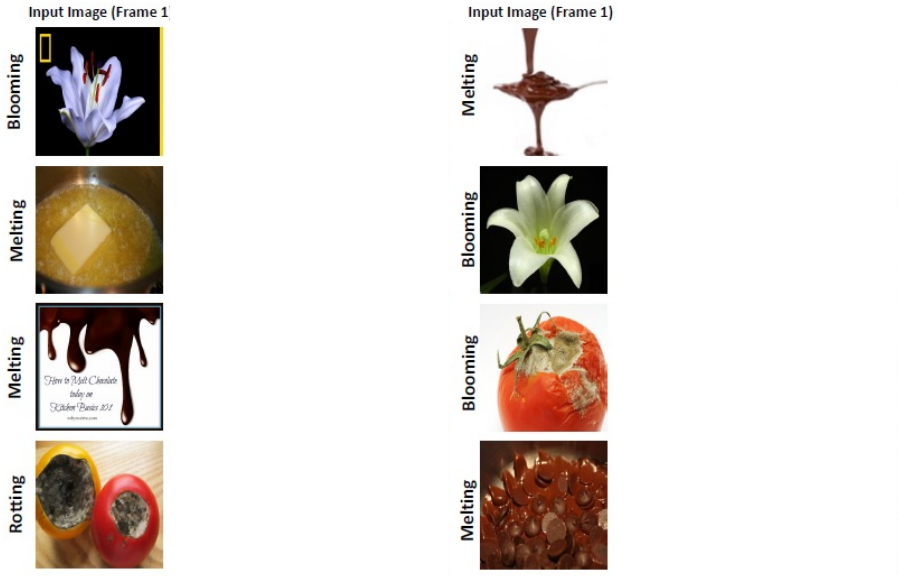


$$\mathcal{L}_{\text{flow}}(\mathbf{y}_{i-1}, \mathbf{y}_i; \mathbf{s}_{i-1}, \mathbf{s}_i) = \sum_l \frac{1}{C_l H_l W_l} \| \underbrace{\Xi(\mathbf{y}_{i-1}, \mathbf{y}_i)_l}_{\text{Optical flow in generated video}} - \underbrace{\Xi(\mathbf{s}_{i-1}, \mathbf{s}_i)_l}_{\text{Optical flow in source video}} \|_2^2$$

Key idea: Generate videos with **similar flow patterns** as source videos (+ many details).

Thomas, Song and Kovashka

Transferring Motion



Thomas, Song and Kovashka

Transferring Motion

Baking

Blooming

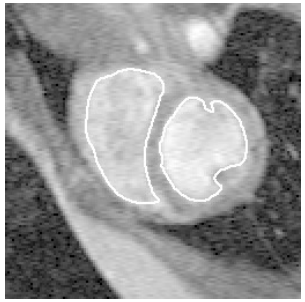
Thomas, Song and Kovashka

Tracking: some applications

Body pose tracking, activity recognition

Censusing a bat population

Video-based interfaces



Medical apps

Surveillance

Kristen Grauman

Tracking examples

Traffic: https://www.youtube.com/watch?v=DiZHQ4peqjg

Soccer: http://www.youtube.com/watch?v=ZqQIItFAnxg

Face: http://www.youtube.com/watch?v=i_bZNVmhJ2o

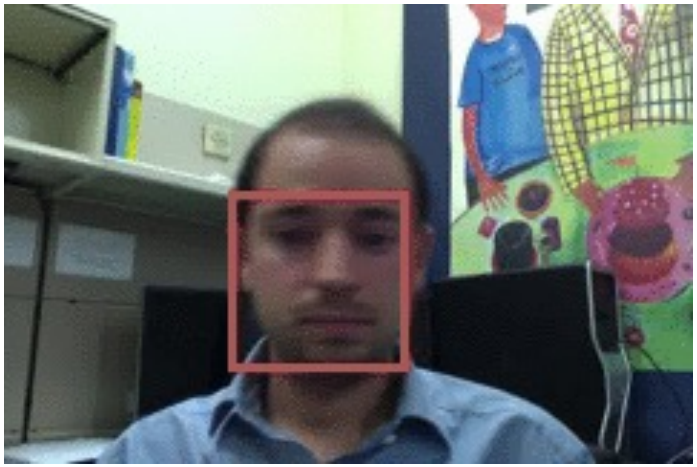
Body: https://www.youtube.com/watch?v=_Ahy0Gh69-M

Eye: http://www.youtube.com/watch?v=NCtYdUEMotg

Gaze: http://www.youtube.com/watch?v=-G6Rw5cU-1c

Things that make visual tracking difficult

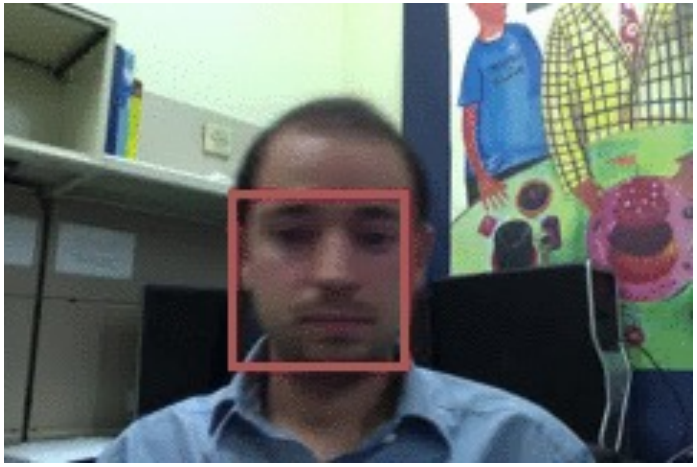
- Erratic movements, moving very quickly
- Occlusions, leaving and coming back
- Surrounding similar-looking objects



Adapted from Amin Sadeghi

Strategies for tracking

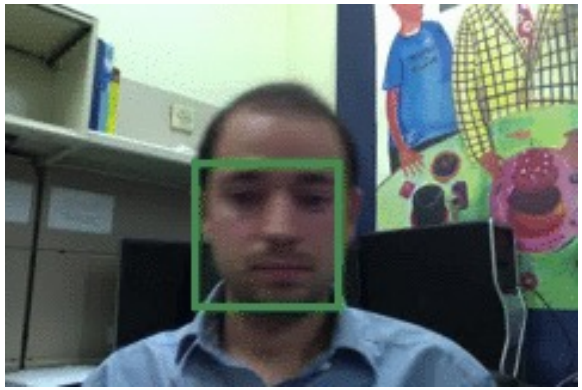
- Tracking by repeated detection
  - Works well if object is easily detectable (e.g., face or colored glove) and there is only one
  - Need some way to link up detections



Amin Sadeghi

Strategies for tracking

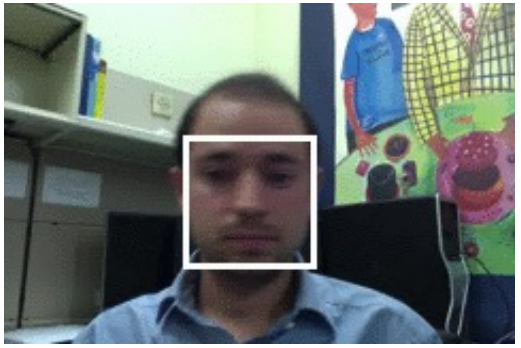
- Tracking w/ dynamics: Using model of expected motion, *predict* object location in next frame
  - Restrict search for the object
  - Measurement noise is reduced by trajectory smoothness
  - Robustness to missing or weak observations
  - Assumptions: Camera is not moving instantly to new viewpoint, objects do not disappear/reappear in different places in the scene



Amin Sadeghi

General model for tracking

- *State* X: The actual state of the moving object that we want to estimate but cannot observe
  - E.g. position, velocity
- *Observations* Y: Our actual measurement or observation of state X, which can be very noisy
- At each time t, the state changes to X_t and we get a new observation Y_t
- Our goal is to recover the most likely state X_t given:
  - All observations so far, i.e. $y_1, y_2, \ldots, y_{t-1}$
  - Knowledge about dynamics of state transitions



Google Colab: link

Adapted from Amin Sadeghi and Kristen Grauman

Steps of tracking

- **Prediction**: What is the next state of the object given *past* measurements?

$$P\left(X_t \middle| Y_0 = y_0, \cdots, Y_{t-1} = y_{t-1}\right)$$

Steps of tracking

- **Prediction**: What is the next state of the object given *past* measurements?

$$P\left(X_t \middle| Y_0 = y_0, \cdots, Y_{t-1} = y_{t-1}\right)$$

- **Correction**: Compute an updated estimate of the state from prediction and measurements

$$P\left(X_t \middle| Y_0 = y_0, \cdots, Y_{t-1} = y_{t-1}, Y_t = y_t\right)$$

Kristen Grauman

Problem statement

- We have models for

Likelihood of next state given current state (dynamics model):

$$P(X_t|X_{t-1})$$

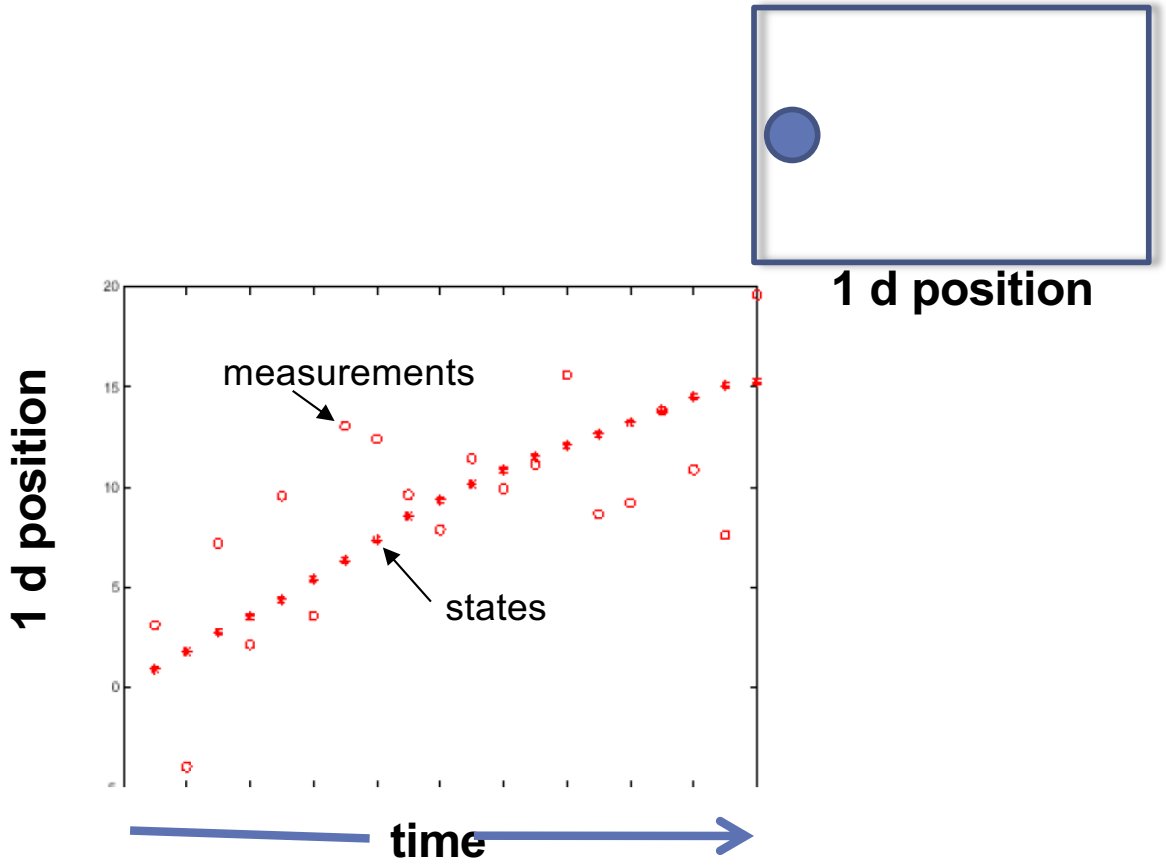
Likelihood of observation given the state (observation or measurement model):

$$P(Y_t|X_t)$$

- We want to recover, for each t:

$$P(X_t|y_0, \cdots, y_t)$$

Example: Constant velocity (1D points)



Kristen Grauman

Example: Constant velocity (1D points)

- State vector: position *p* and velocity *v*

$$x_t = \begin{bmatrix} p_t \\ v_t \end{bmatrix} \qquad \begin{aligned} p_t &= p_{t-1} + (\Delta t)v_{t-1} + \varepsilon \\ v_t &= v_{t-1} + \xi \end{aligned}$$

$$x_t = D_t x_{t-1} + noise = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_{t-1} \\ v_{t-1} \end{bmatrix} + noise$$

- Measurement is position only

$$y_t = M x_t + noise = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} p_t \\ v_t \end{bmatrix} + noise$$

Prediction and correction

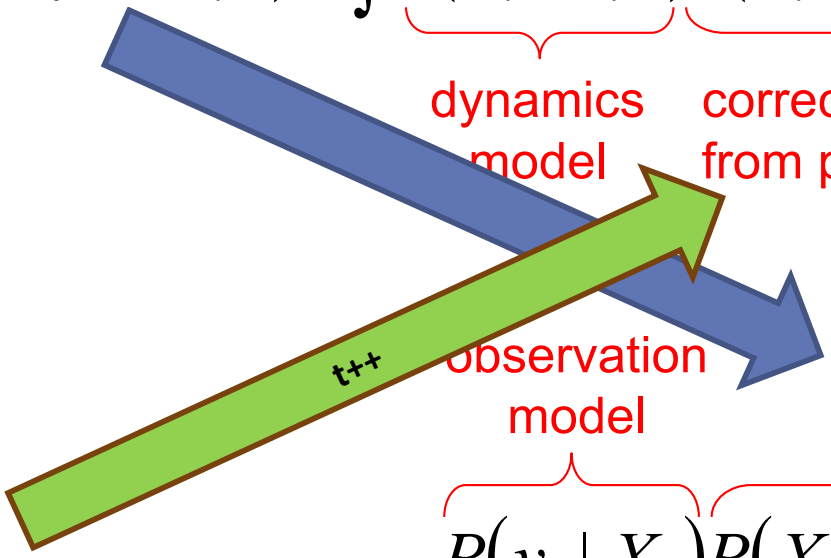
Prediction:

$$P(X_t \mid y_0, \ldots, y_{t-1}) = \int P(X_t \mid X_{t-1}) P(X_{t-1} \mid y_0, \ldots, y_{t-1}) dX_{t-1}$$

dynamics model

corrected estimate from previous step

Correction:



observation model

predicted estimate

$$P(X_t \mid y_0, \ldots, y_t) = \frac{P(y_t \mid X_t) P(X_t \mid y_0, \ldots, y_{t-1})}{\int P(y_t \mid X_t) P(X_t \mid y_0, \ldots, y_{t-1}) dX_t}$$

Adapted from Amin Sadeghi