

CS 1674: Sequential Data: Language and Vision

PhD. Nils Murrugarra-Llerena
nem177@pitt.edu



Plan for this lecture

- Language and vision
 - Application: Image captioning
 - Tool: Recurrent neural networks
 - Tool: Transformers
 - Application: Visual question answering



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.



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.



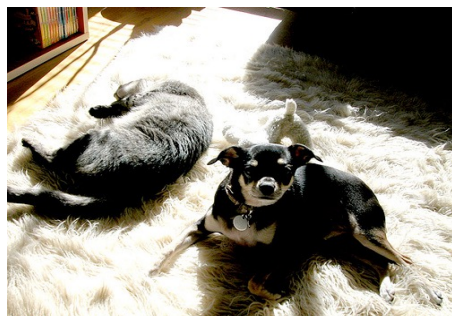
This is a picture of two dogs. The first dog is near the second furry dog.

Some pre-RNN Good Results

Missed detections:



Here we see one potted plant.



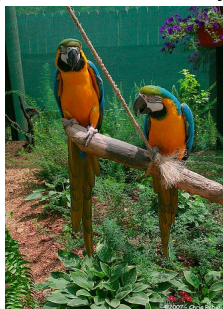
This is a picture of one dog.

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 sheep 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."

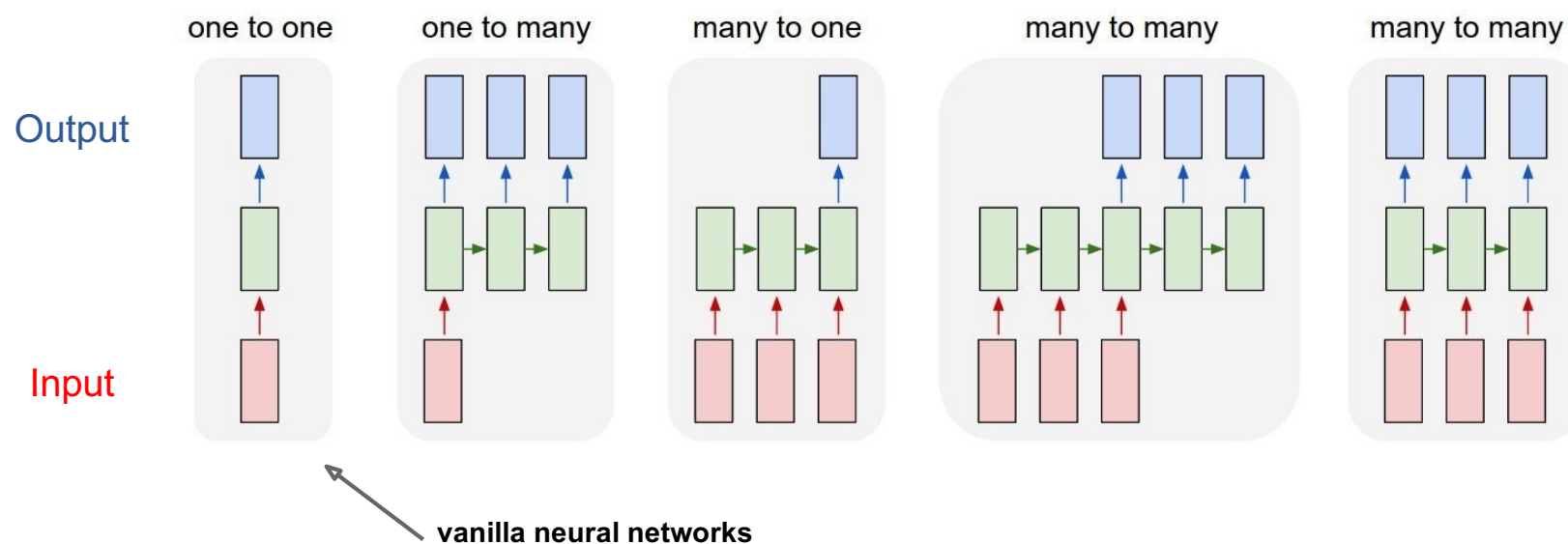


"two young girls are playing with lego toy."

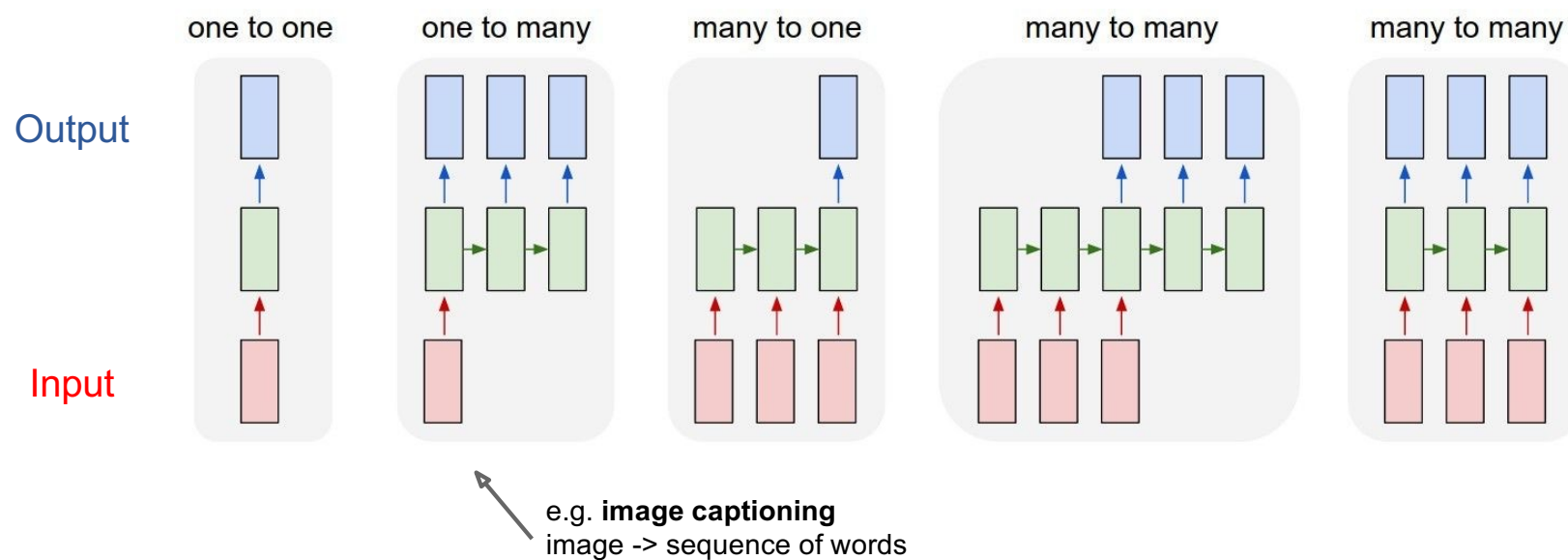


"boy is doing backflip on wakeboard."

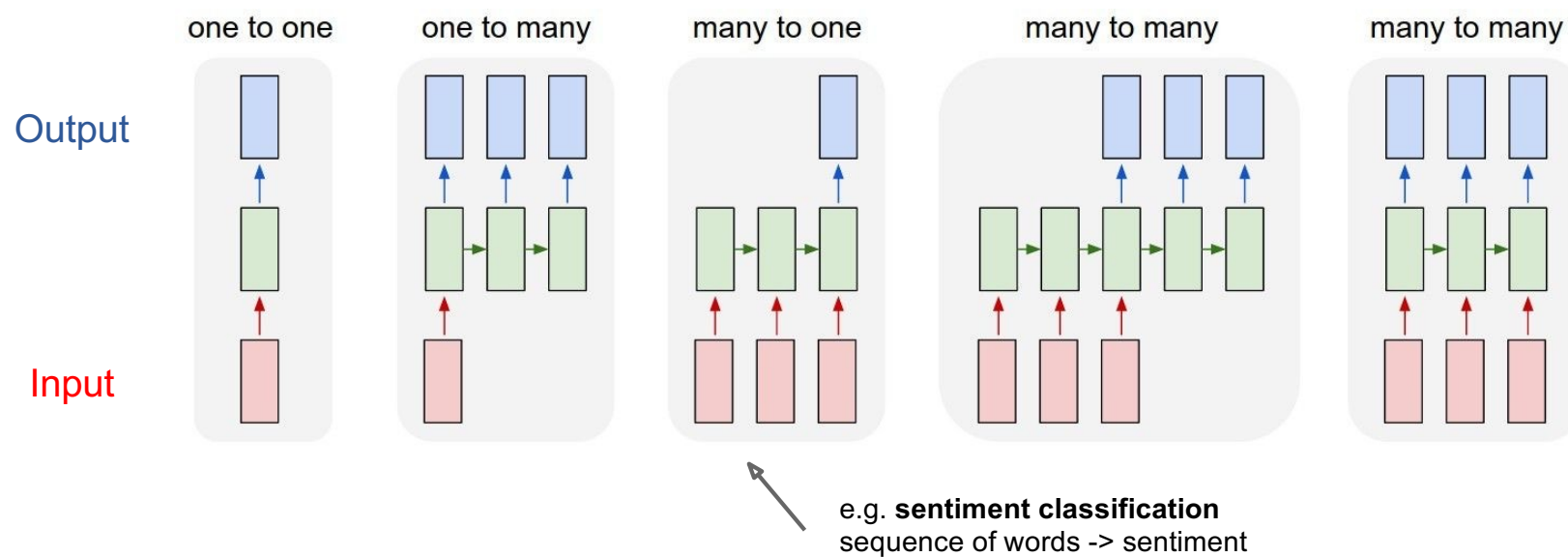
Recurrent Networks offer a lot of flexibility:



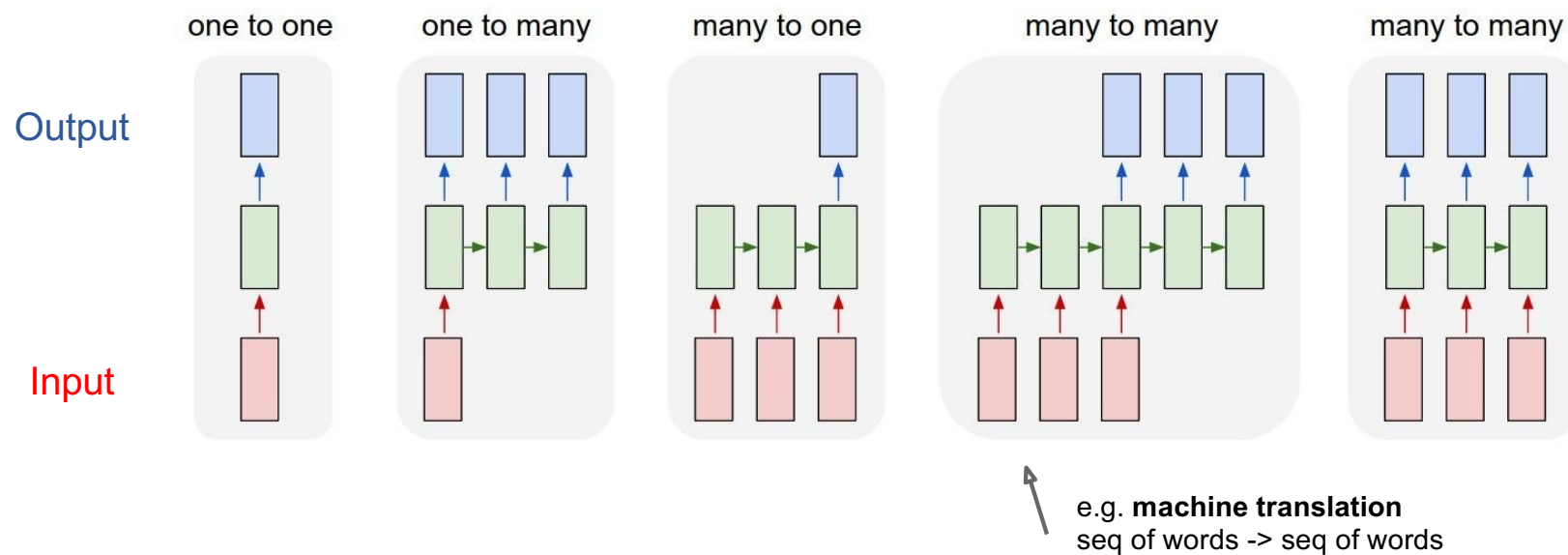
Recurrent Networks offer a lot of flexibility:



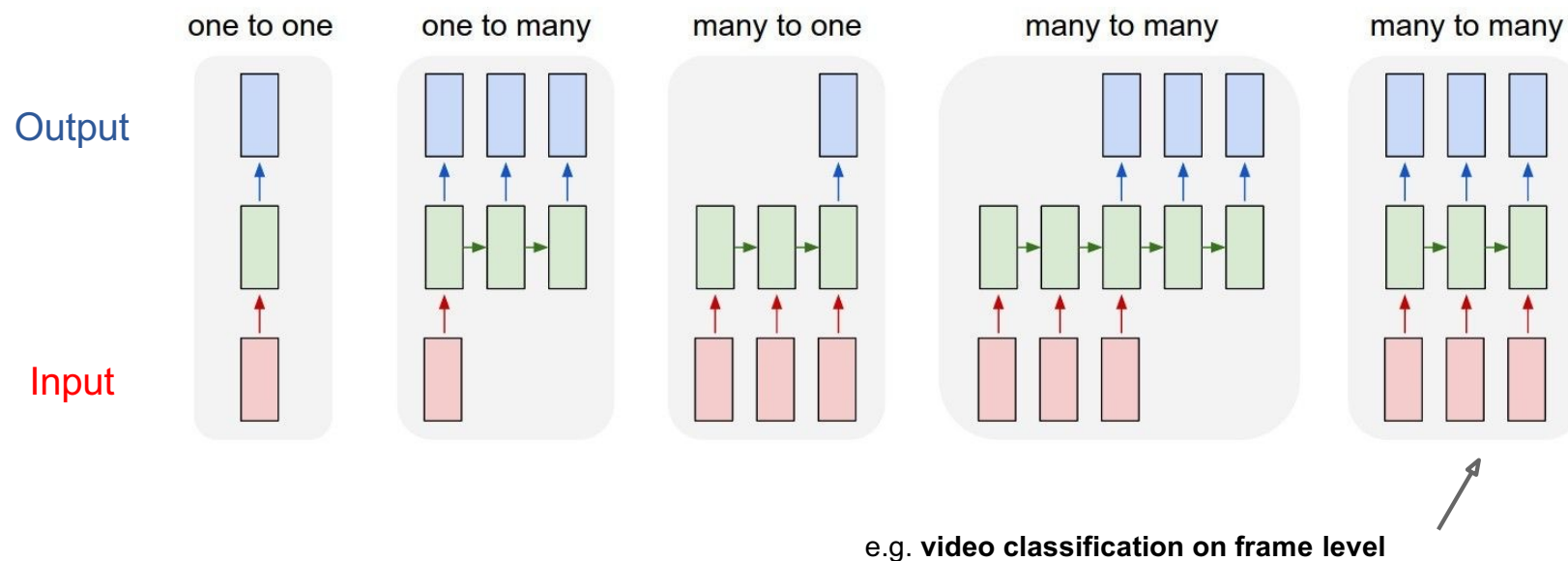
Recurrent Networks offer a lot of flexibility:



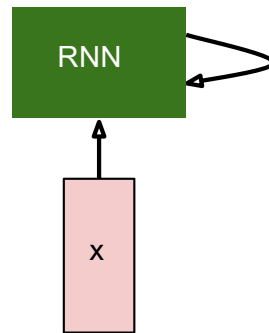
Recurrent Networks offer a lot of flexibility:



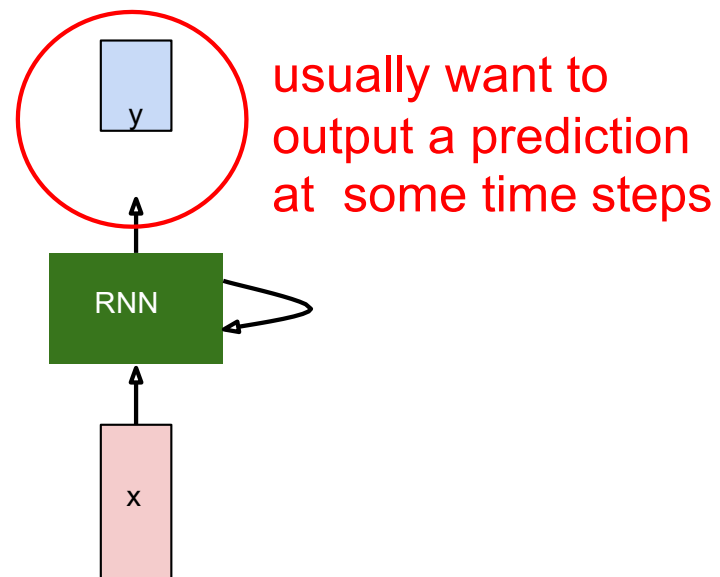
Recurrent Networks offer a lot of flexibility:



Recurrent Neural Network



Recurrent Neural Network



Adapted from Andrej Karpathy

Recurrent Neural Network

We can process a sequence of vectors \mathbf{x} by applying a recurrence formula at every time step:

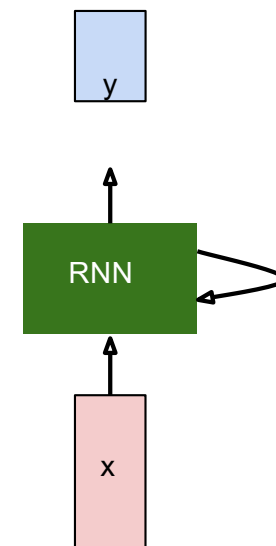
$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

some function with parameters W

old state

input vector at some time step

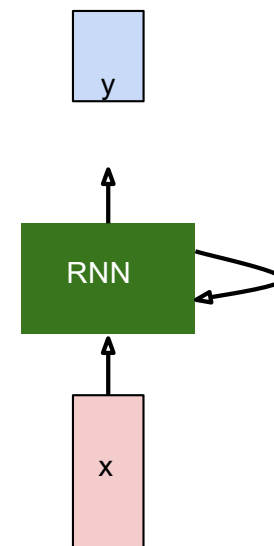


Recurrent Neural Network

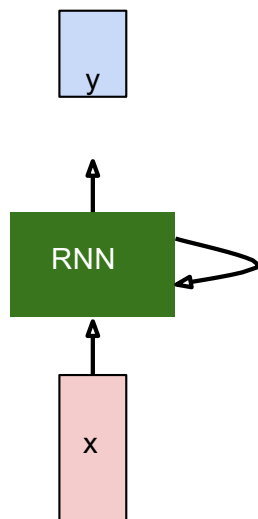
We can process a sequence of vectors \mathbf{x} by applying a recurrence formula at every time step:

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

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



(Vanilla) Recurrent Neural Network



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



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

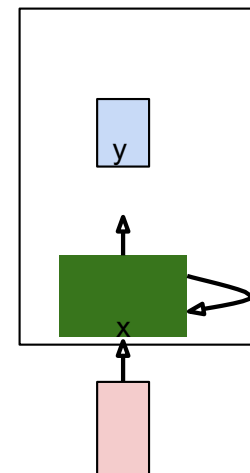
$$y_t = W_{hy}h_t$$

Example

Character-level language model example

Vocabulary:
[h,e,l,o]

Example training
sequence:
“hello”

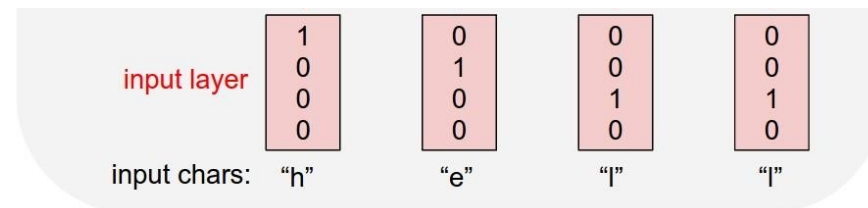


Example

Character-level language model example

Vocabulary:
[h,e,l,o]

Example training
sequence:
“hello”



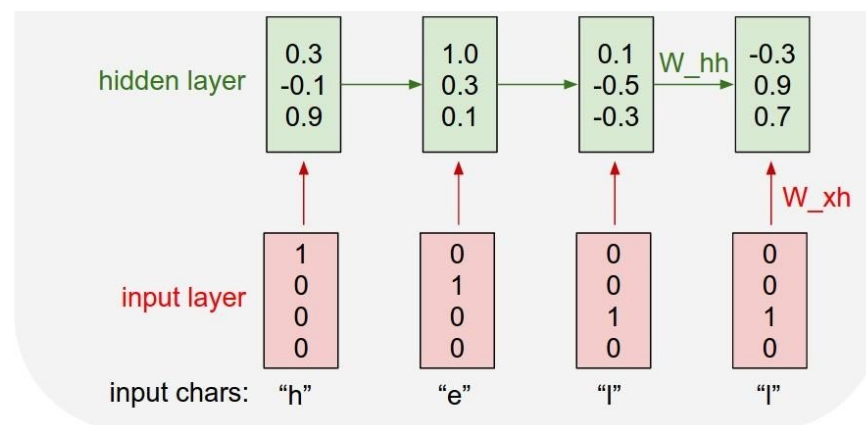
Example

Character-level language model example

Vocabulary:
[h,e,l,o]

Example training
sequence:
“hello”

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

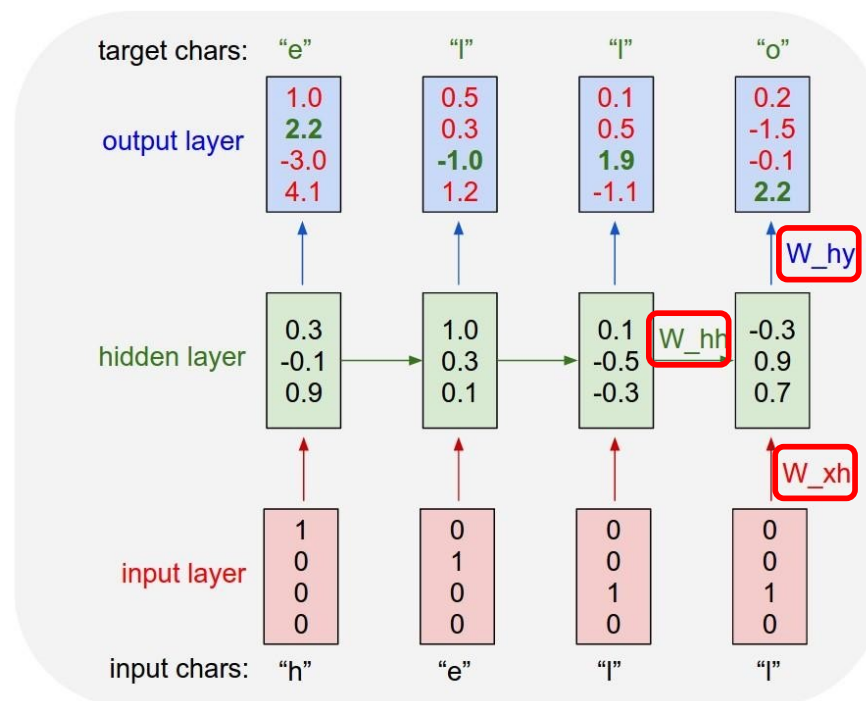


Example

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 tklrge t o idoe ns,smtt h ne etie h,hregtrs niglike,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 oftended 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/>

Generating poetry with RNNS

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 nudes 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 apt, 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.

Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models)

Translations: Chinese (Simplified), Japanese, Korean, Russian

Watch: MIT's [Deep Learning State of the Art](#) lecture referencing this post

May 25th update: New graphics (RNN animation, word embedding graph), color coding, elaborated on the final attention example.

Note: The animations below are videos. Touch or hover on them (if you're using a mouse) to get play controls so you can pause if needed.

Sequence-to-sequence models are deep learning models that have achieved a lot of success in tasks like machine translation, text summarization, and image captioning. Google Translate started [using](#) such a model in production in late 2016. These models are explained in the two pioneering papers ([Sutskever et al., 2014](#), [Cho et al., 2014](#)).



<https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/>

<https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/>

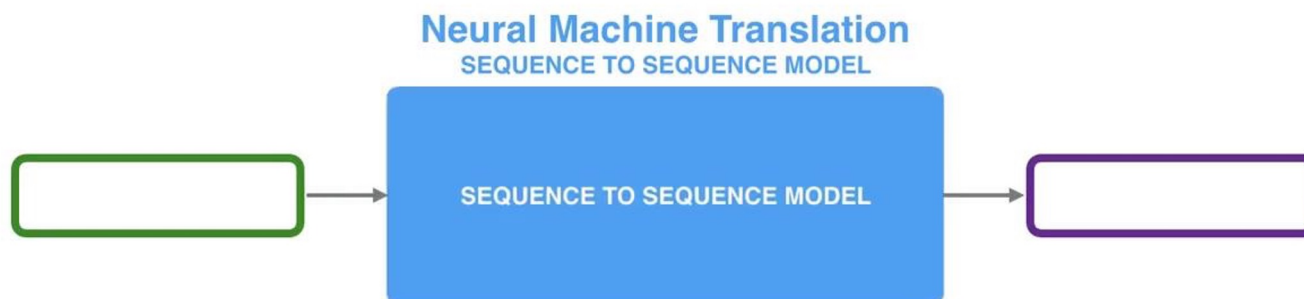
Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models)

Translations: Chinese (Simplified), Japanese, Korean, Russian

Watch: MIT's [Deep Learning State of the Art](#) lecture referencing this post

May 25th update: New graphics (RNN animation, word embedding graph), color coding, elaborated on the final attention example.



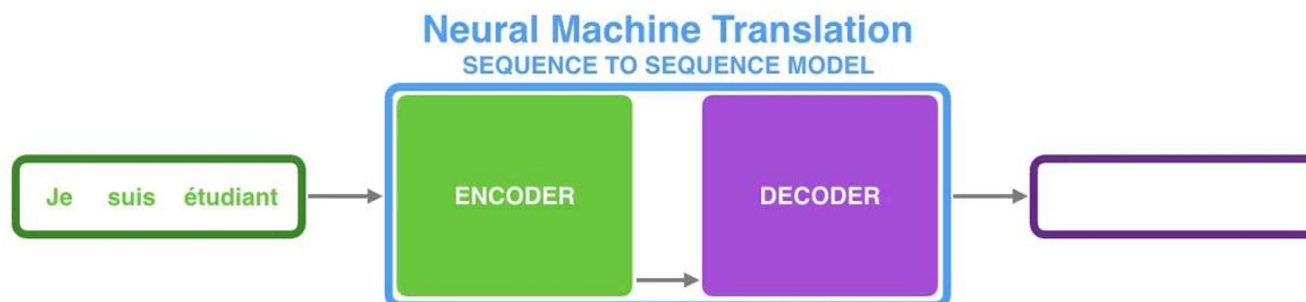
<https://falammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/>

Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models)

Translations: Chinese (Simplified), Japanese, Korean, Russian

Watch: MIT's [Deep Learning State of the Art](#) lecture referencing this post

May 25th update: New graphics (RNN animation, word embedding graph), color coding, elaborated on the final attention example.



<https://falammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/>

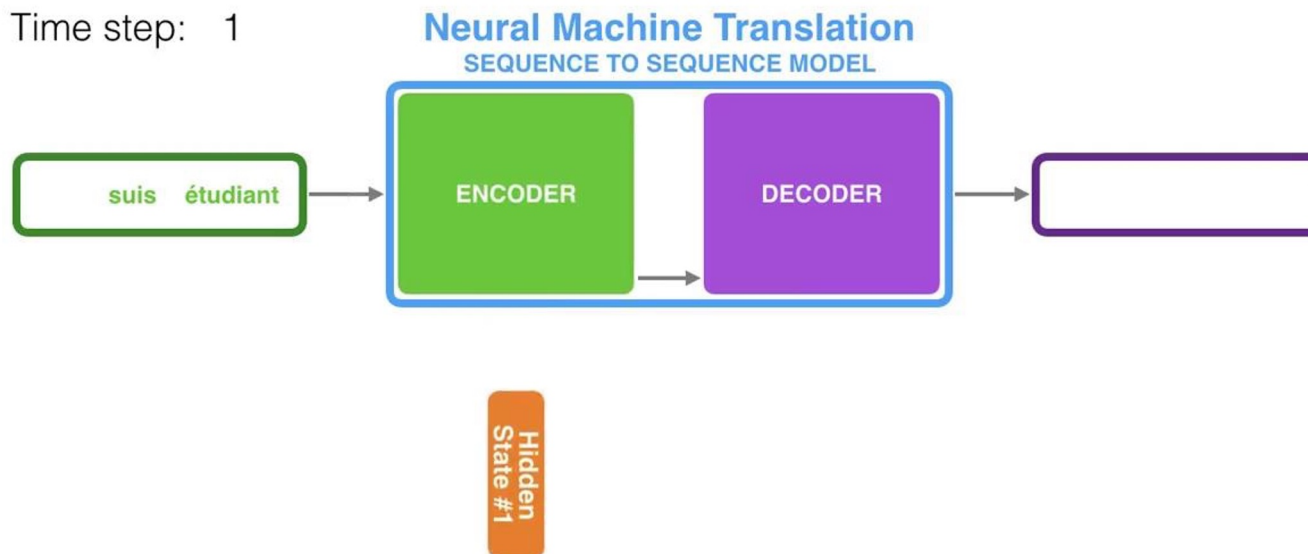
Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models)

Translations: Chinese (Simplified), Japanese, Korean, Russian

Watch: MIT's [Deep Learning State of the Art](#) lecture referencing this post

May 25th update: New graphics (RNN animation, word embedding graph), color coding, elaborated on the final attention example.

Time step: 1



<https://alammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/>

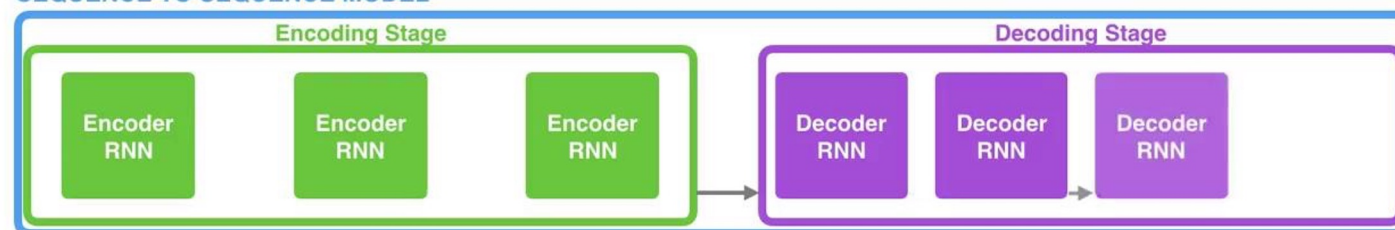
Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models)

Translations: Chinese (Simplified), Japanese, Korean, Russian

Watch: MIT's [Deep Learning State of the Art](#) lecture referencing this post

May 25th update: New graphics (RNN animation, word embedding graph), color coding, elaborated on the final attention example.

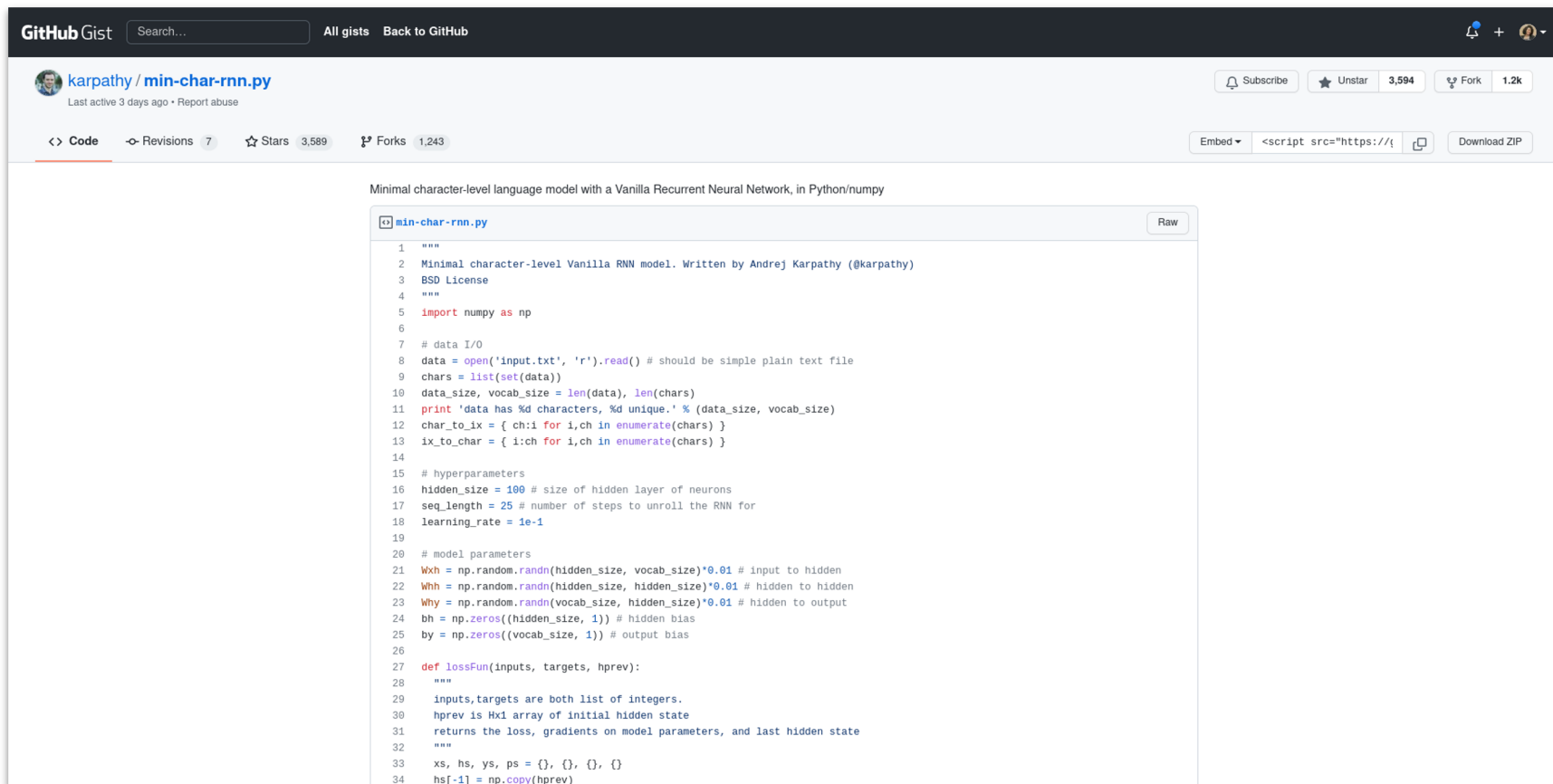
Neural Machine Translation SEQUENCE TO SEQUENCE MODEL



<https://falammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/>

<https://gist.github.com/karpathy/d4dee566867f8291f086> (Andrej Karpathy)

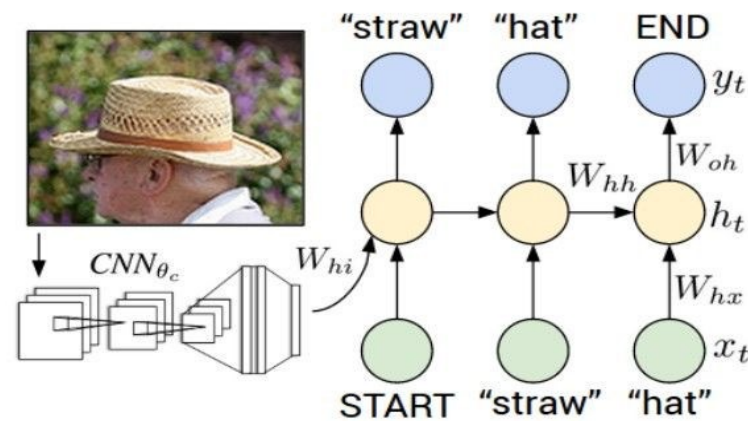
RNN Vanilla: 112 lines of Python



Minimal character-level language model with a Vanilla Recurrent Neural Network, in Python/numpy

```
1 """
2 Minimal character-level Vanilla RNN model. Written by Andrej Karpathy (@karpathy)
3 BSD License
4 """
5 import numpy as np
6
7 # data I/O
8 data = open('input.txt', 'r').read() # should be simple plain text file
9 chars = list(set(data))
10 data_size, vocab_size = len(data), len(chars)
11 print 'data has %d characters, %d unique.' % (data_size, vocab_size)
12 char_to_ix = { ch:i for i,ch in enumerate(chars) }
13 ix_to_char = { i:ch for i,ch in enumerate(chars) }
14
15 # hyperparameters
16 hidden_size = 100 # size of hidden layer of neurons
17 seq_length = 25 # number of steps to unroll the RNN for
18 learning_rate = 1e-1
19
20 # model parameters
21 Wxh = np.random.randn(hidden_size, vocab_size)*0.01 # input to hidden
22 Whh = np.random.randn(hidden_size, hidden_size)*0.01 # hidden to hidden
23 Why = np.random.randn(vocab_size, hidden_size)*0.01 # hidden to output
24 bh = np.zeros((hidden_size, 1)) # hidden bias
25 by = np.zeros((vocab_size, 1)) # output bias
26
27 def lossFun(inputs, targets, hprev):
28     """
29     inputs, targets are both list of integers.
30     hprev is Hx1 array of initial hidden state
31     returns the loss, gradients on model parameters, and last hidden state
32     """
33     xs, hs, ys, ps = {}, {}, {}, {}
34     hs[-1] = np.copy(hprev)
```

Image Captioning



CVPR 2015:

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

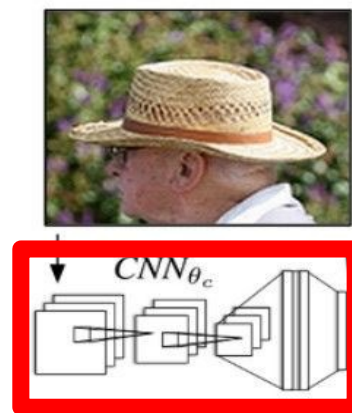
Show and Tell: A Neural Image Caption Generator, Vinyals et al.

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

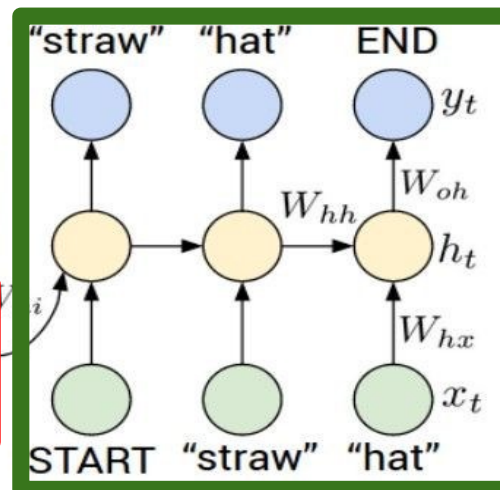
Learning a Recurrent Visual Representation for Image Caption Generation, Chen and Zitnick

Adapted from Andrej Karpathy

Image Captioning



Recurrent Neural Network



Convolutional Neural Network

Image Captioning



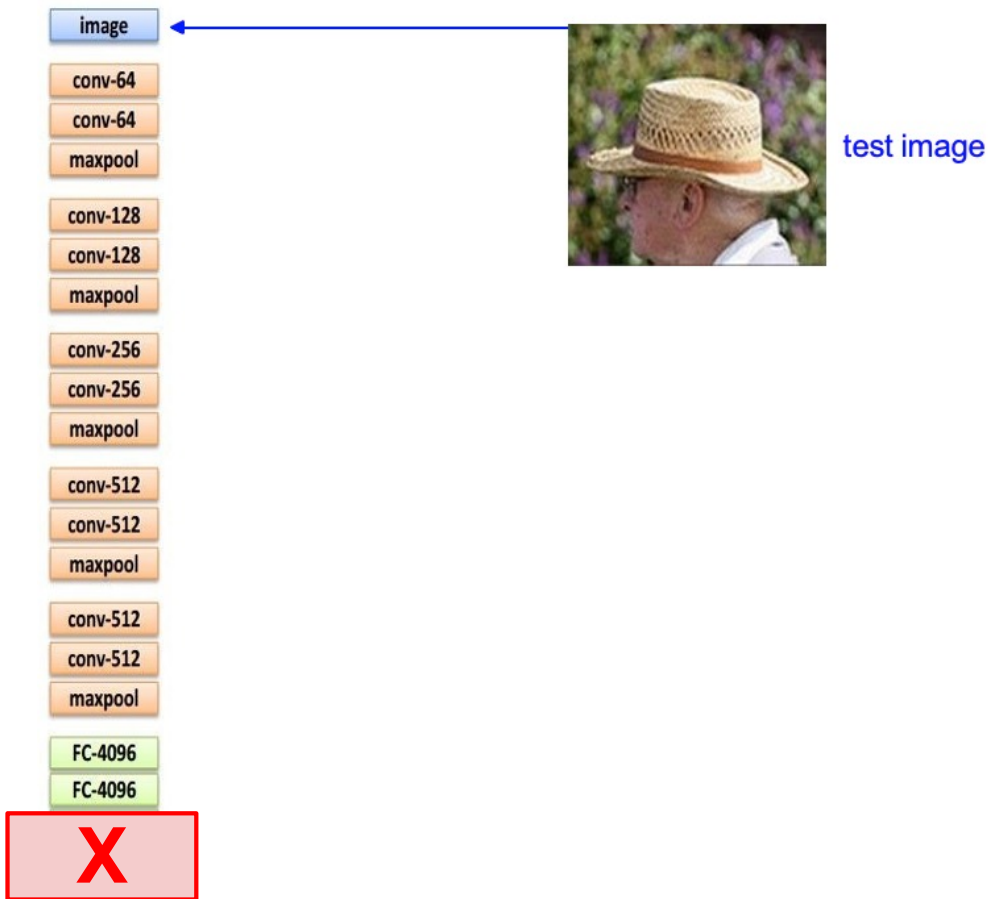
test image

Image Captioning



Andrej Karpathy

Image Captioning



Andrej Karpathy

Image Captioning

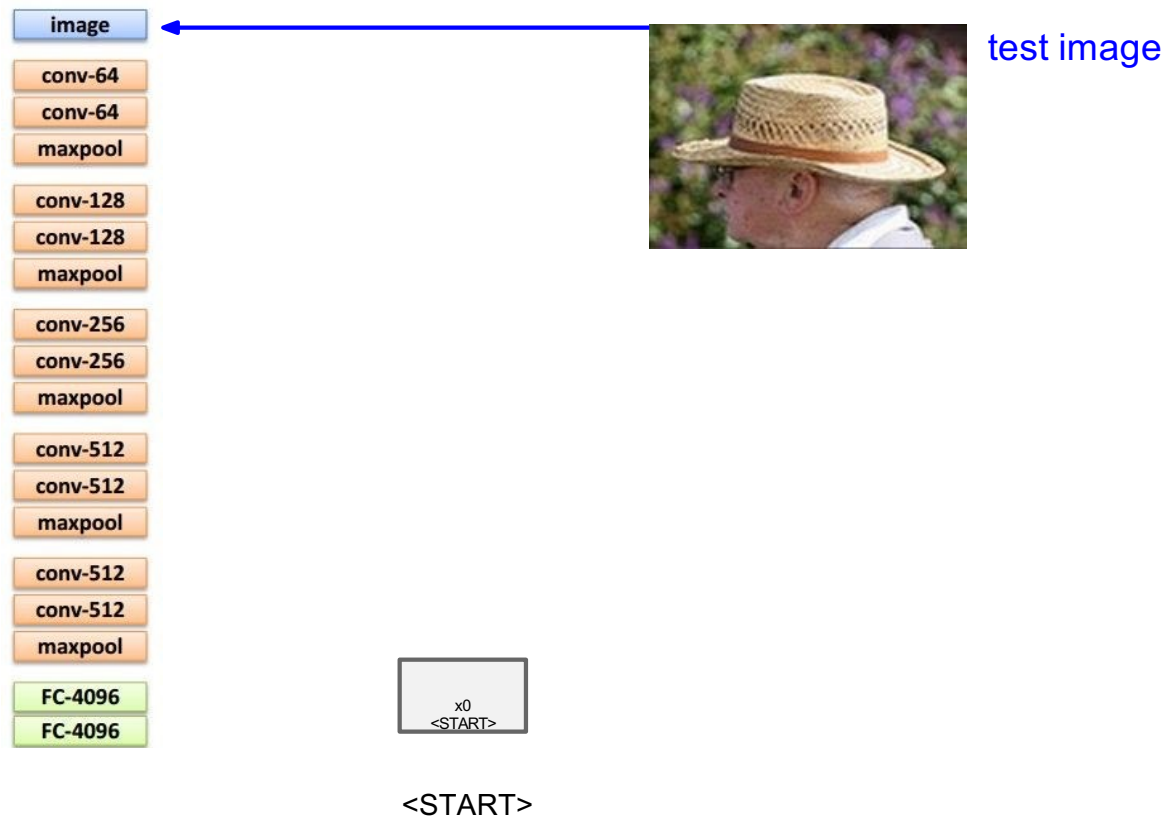
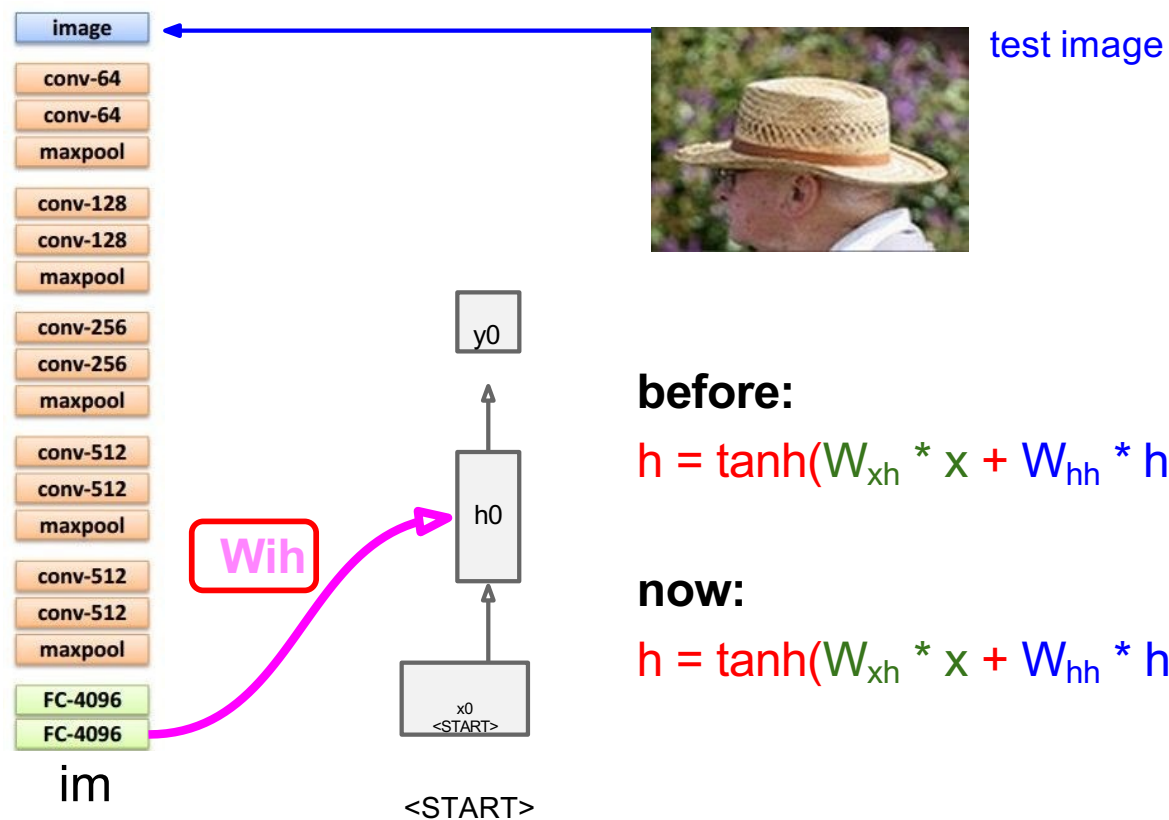


Image Captioning



before:

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

now:

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

Image Captioning

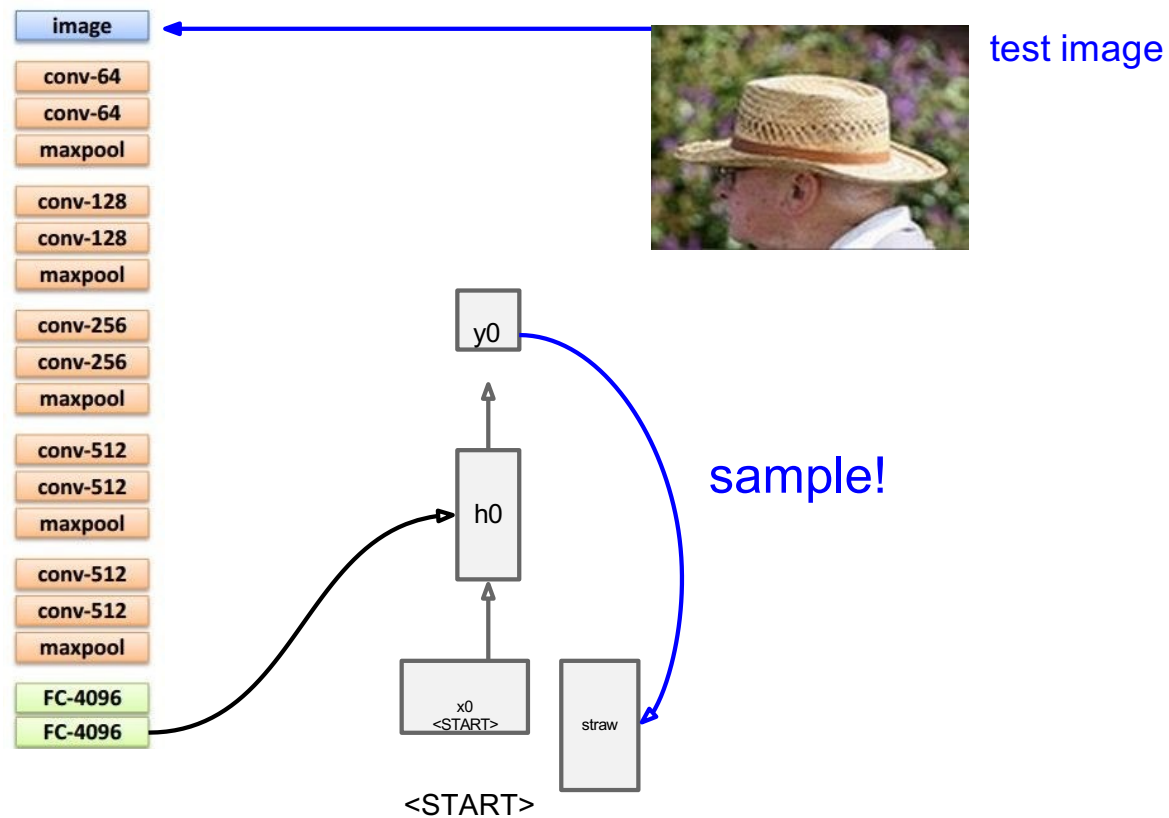


Image Captioning

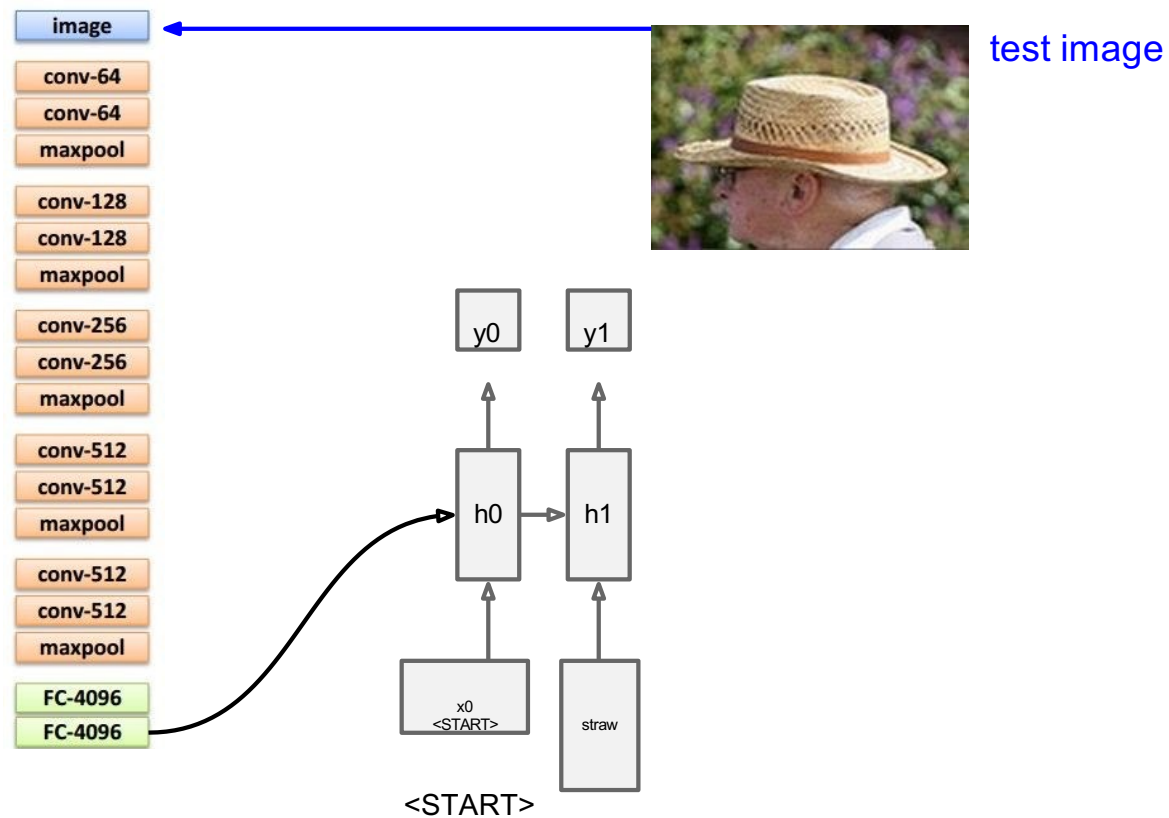


Image Captioning

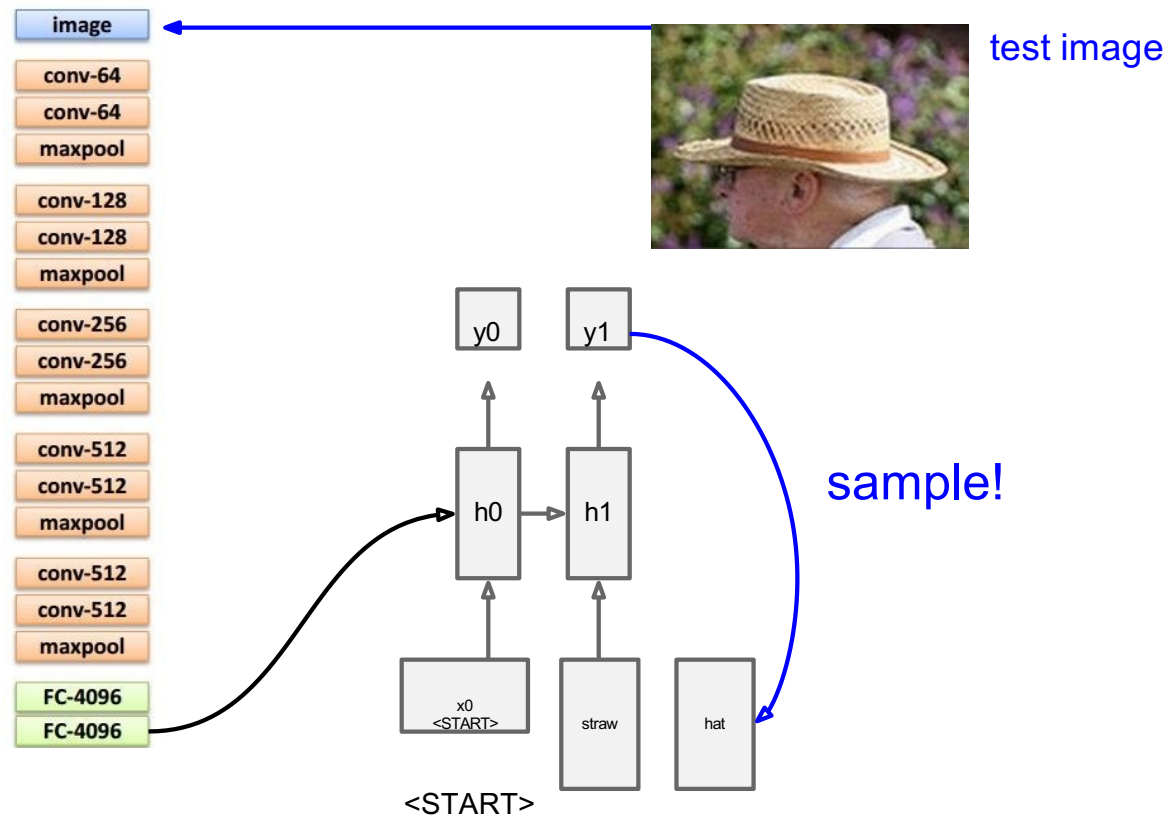


Image Captioning

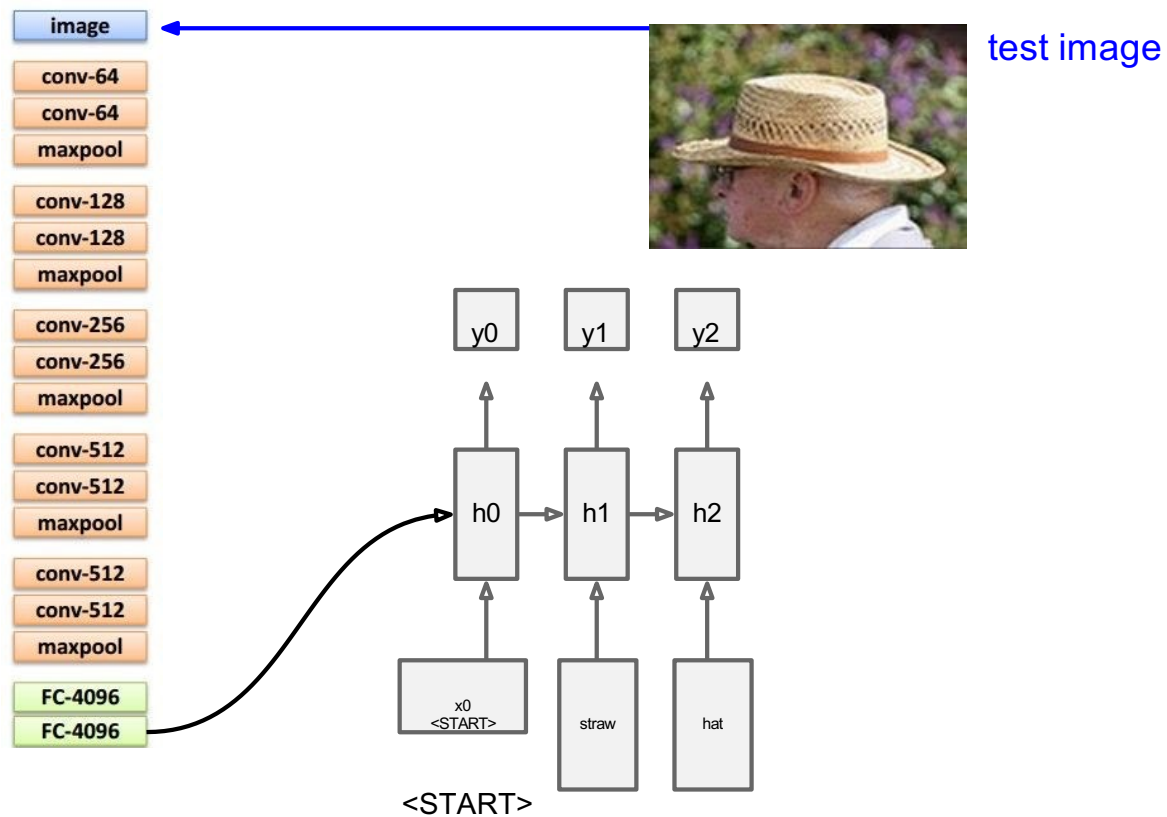
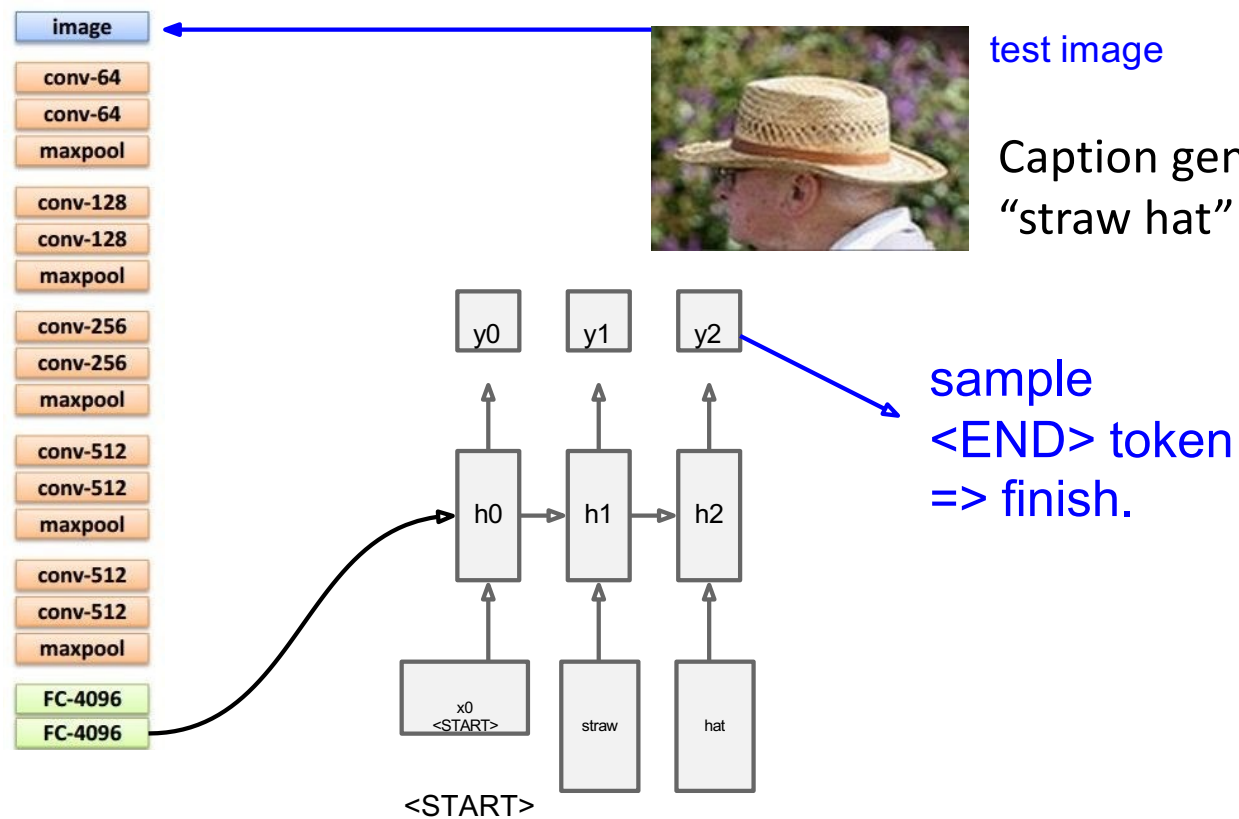


Image Captioning



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."



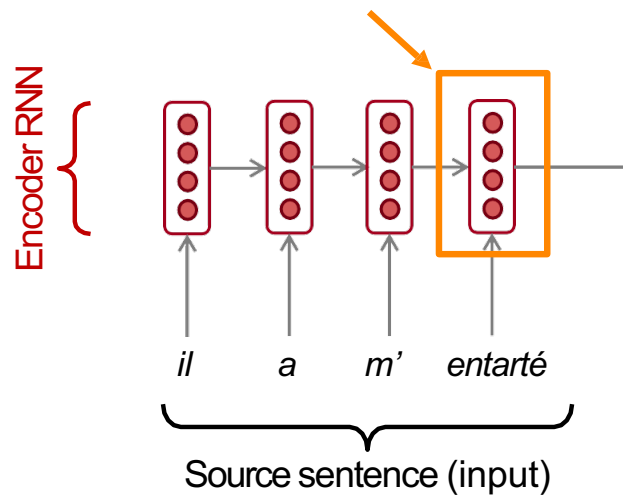
"a horse is standing in the middle of a road."

Andrej Karpathy

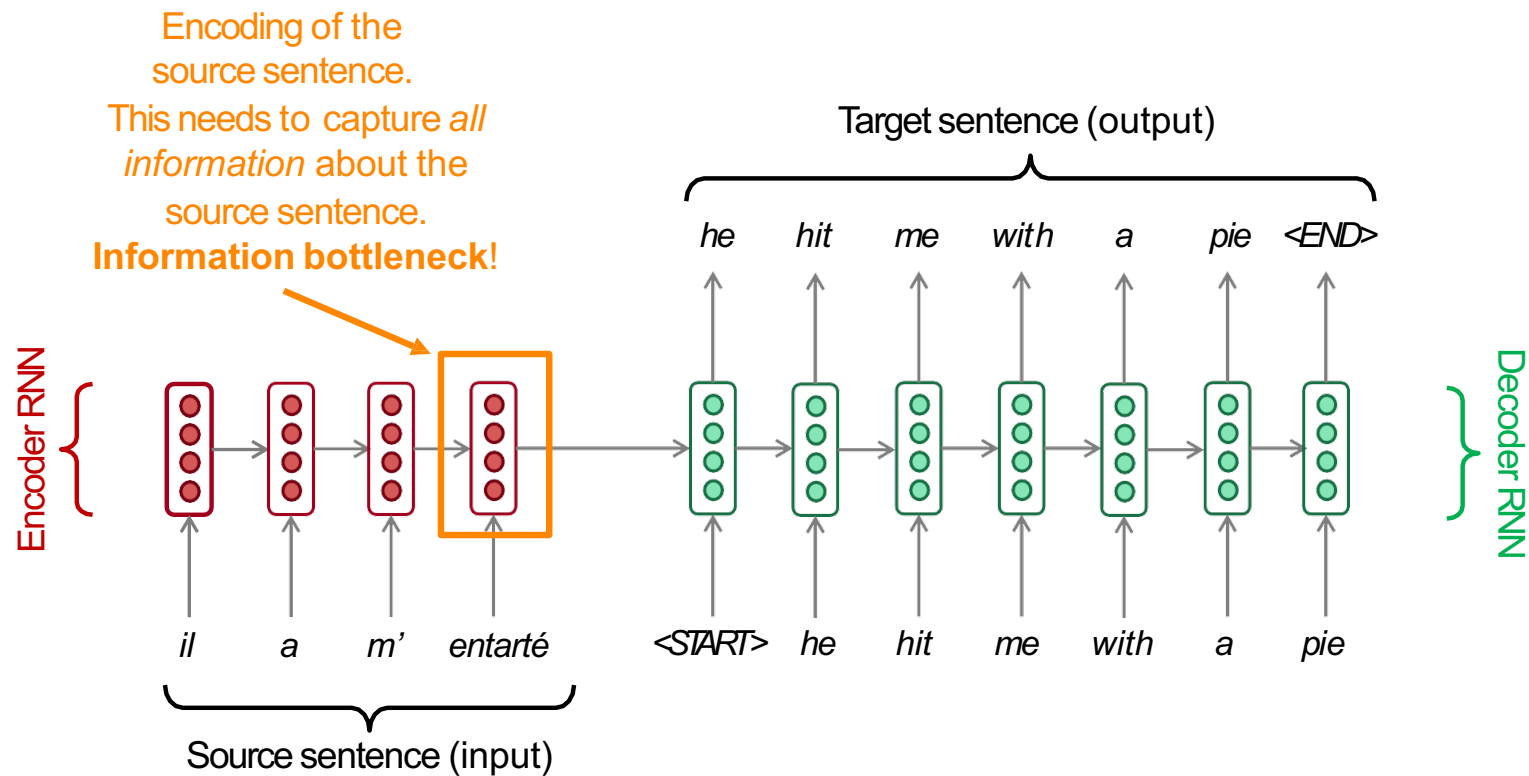
Neural Machine Translation (NMT)

The sequence-to-sequence model

Encoding of the source sentence.
Provides initial hidden state
for Decoder RNN.

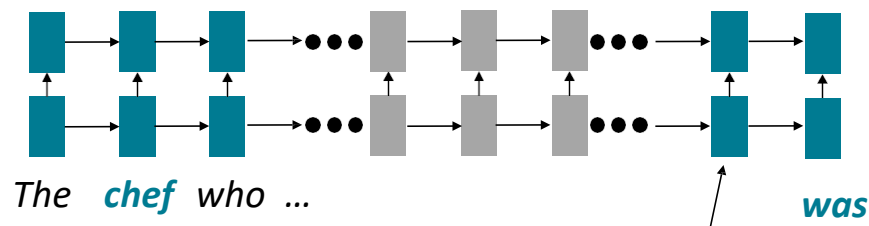


Sequence-to-sequence: the bottleneck problem



Issues with recurrent models: Linear interaction distance

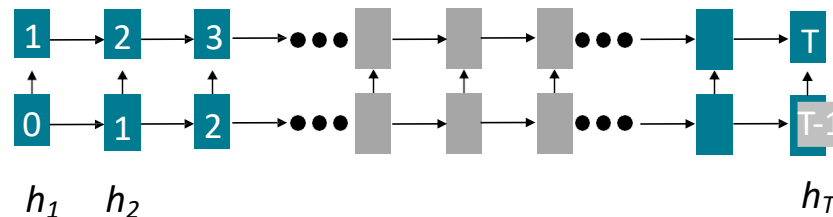
- **$O(\text{sequence length})$** steps for distant word pairs to interact means:
 - Hard to learn long-distance dependencies (because gradient problems!)
 - Linear order of words is “baked in”; not necessarily the right way to think about sentences...



Info of *chef* has gone through $O(\text{sequence length})$ many layers!

Issues with recurrent models: Lack of parallelization

- Forward and backward passes have **$O(\text{sequence length})$** 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

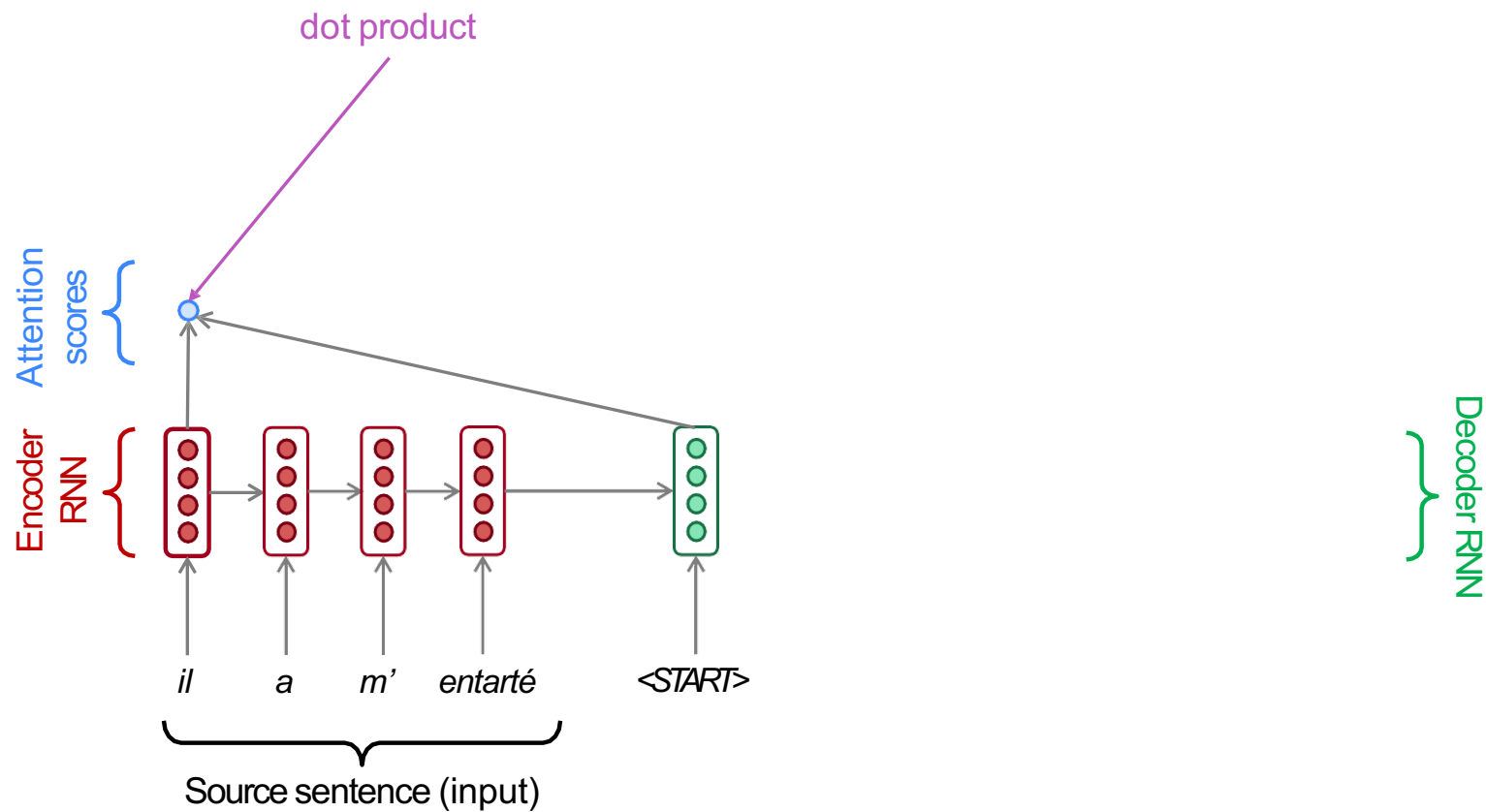
Attention

- **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

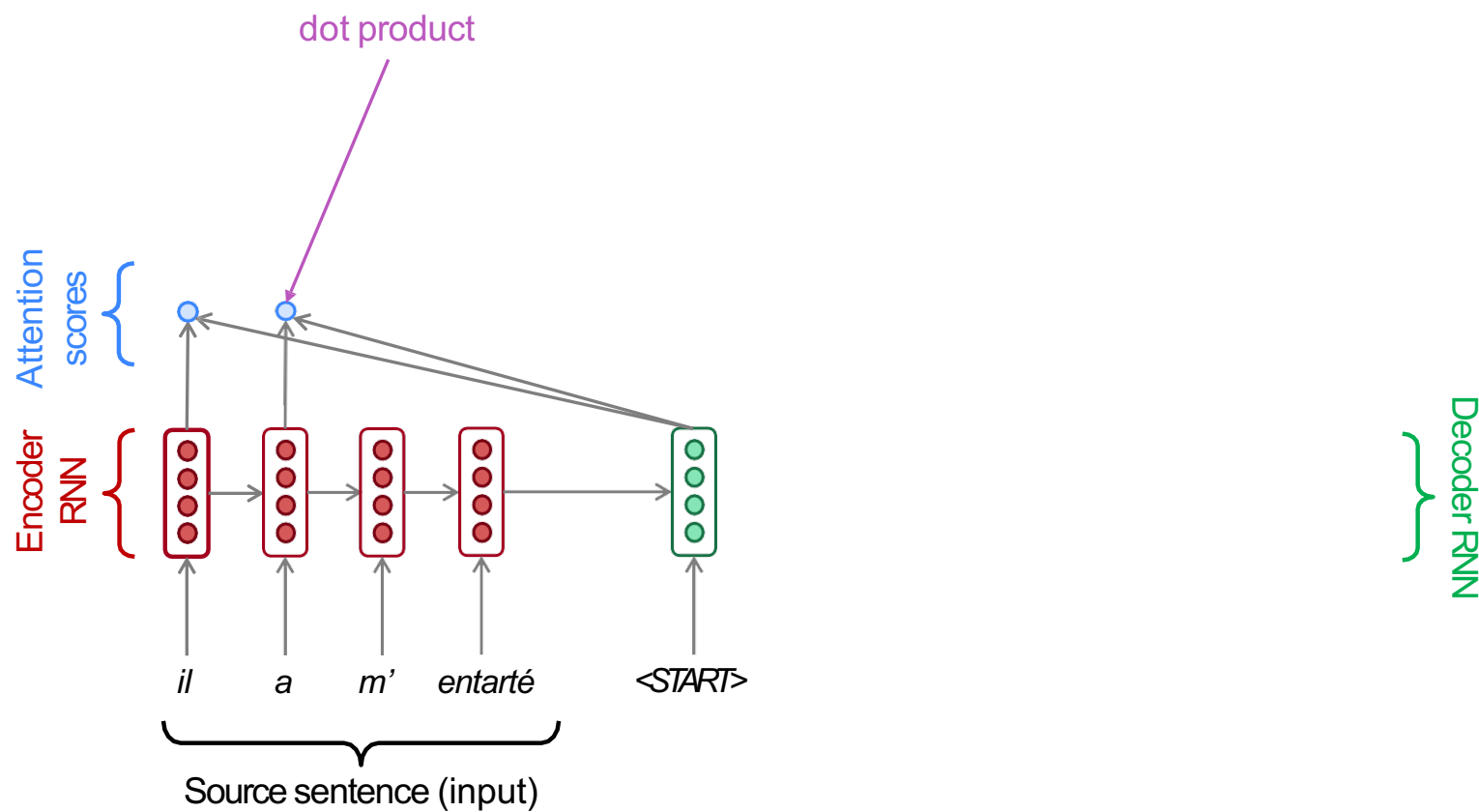


- First, we will show via diagram (no equations), then we will show with equations

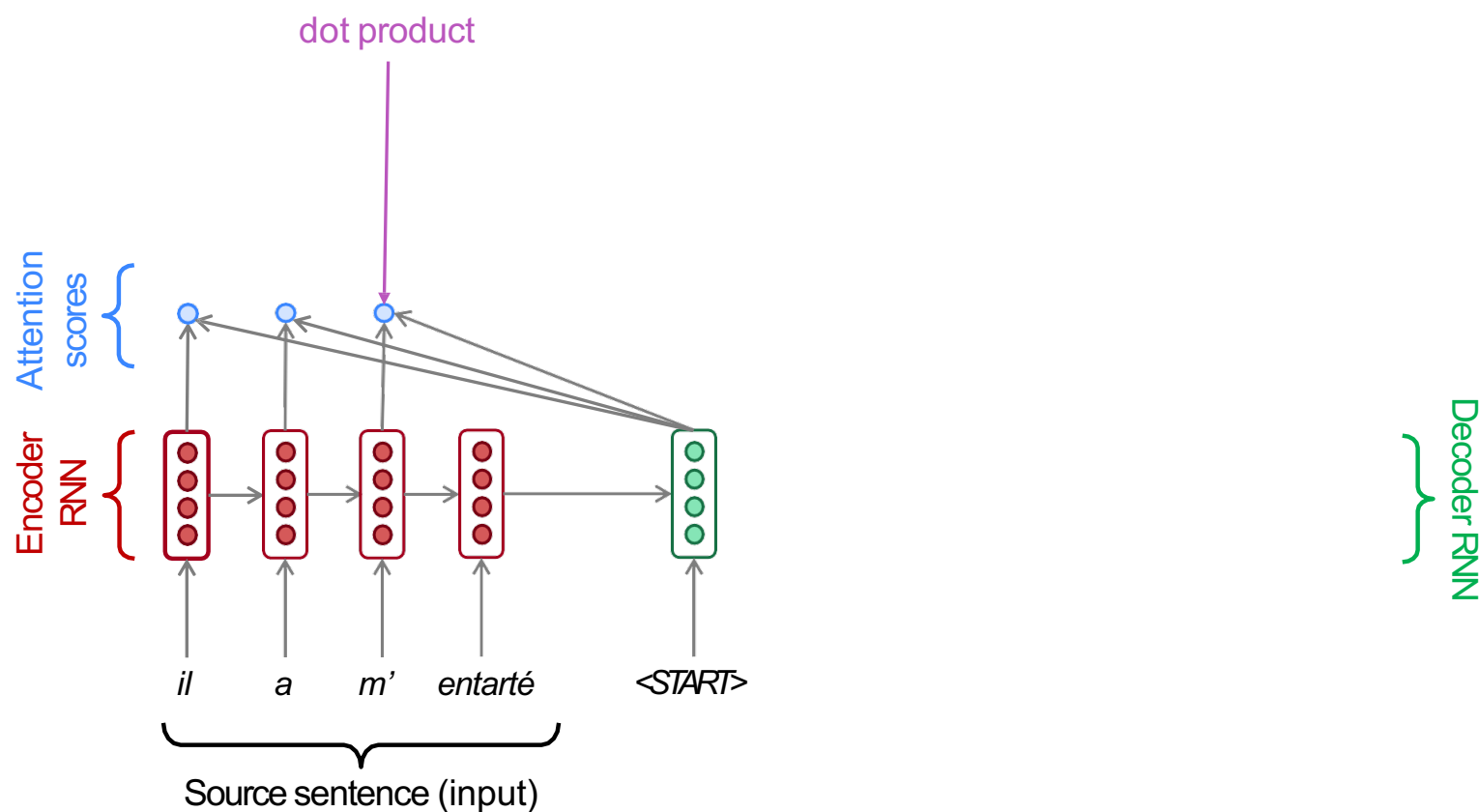
Sequence-to-sequence with attention



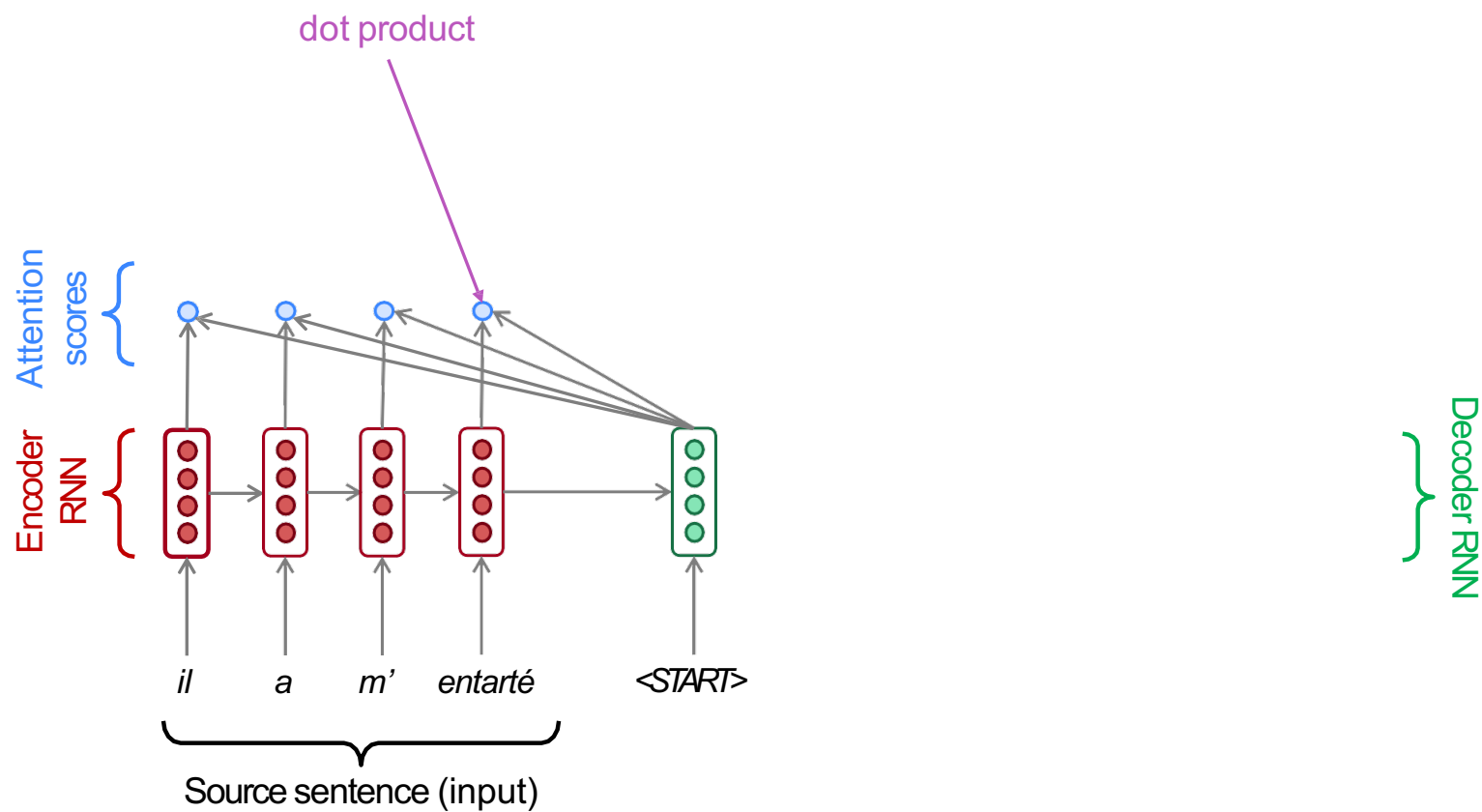
Sequence-to-sequence with attention



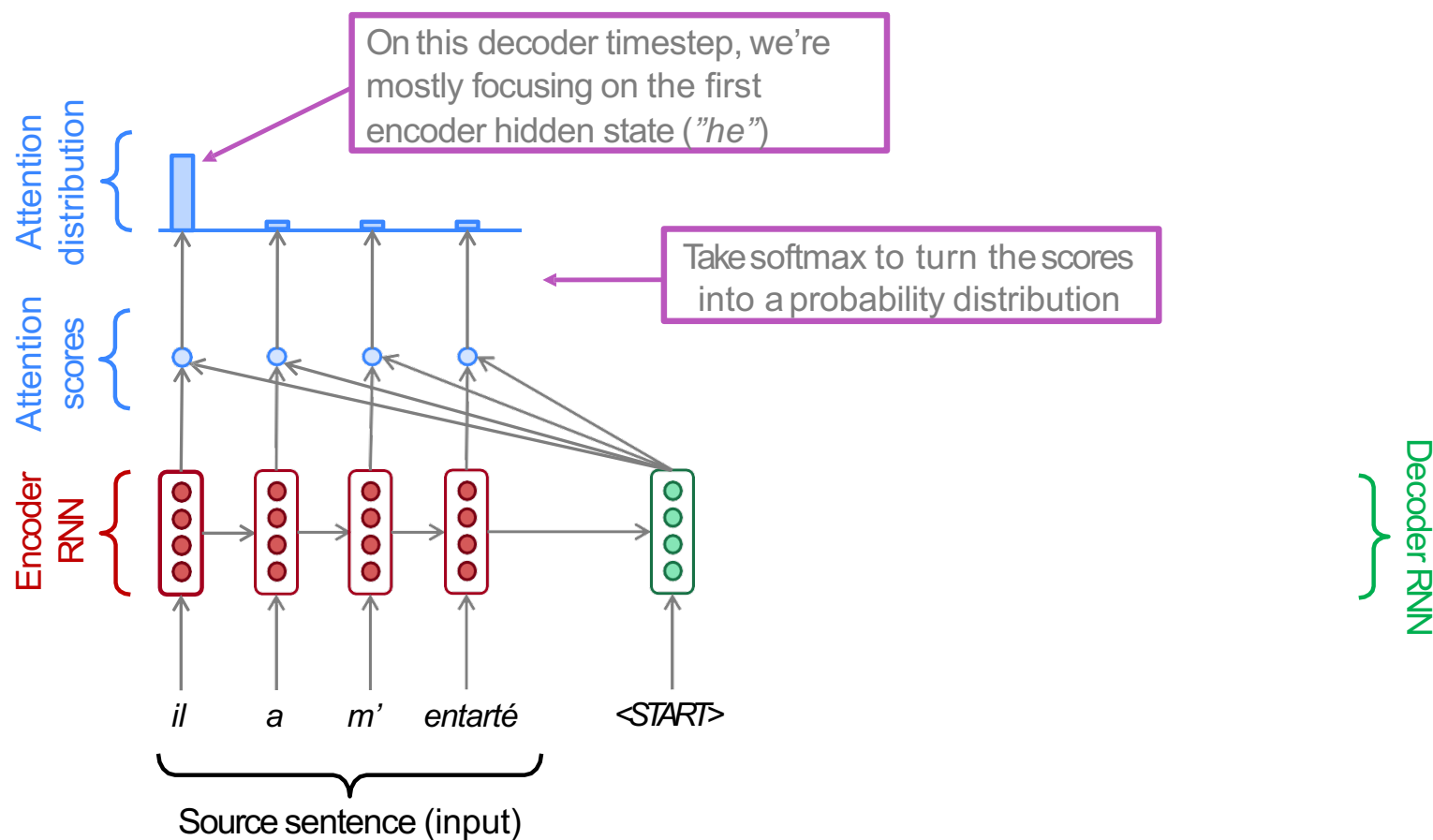
Sequence-to-sequence with attention



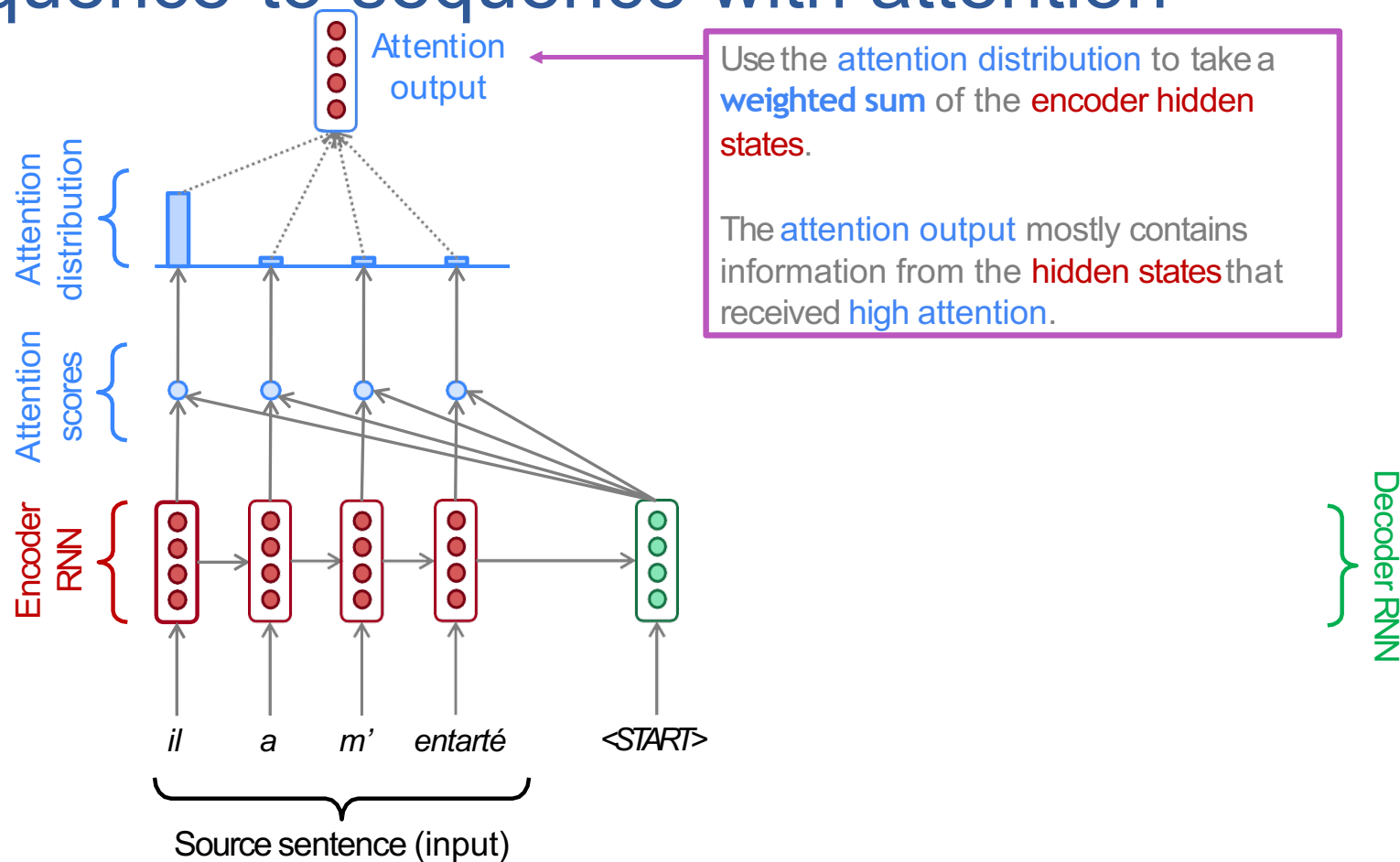
Sequence-to-sequence with attention



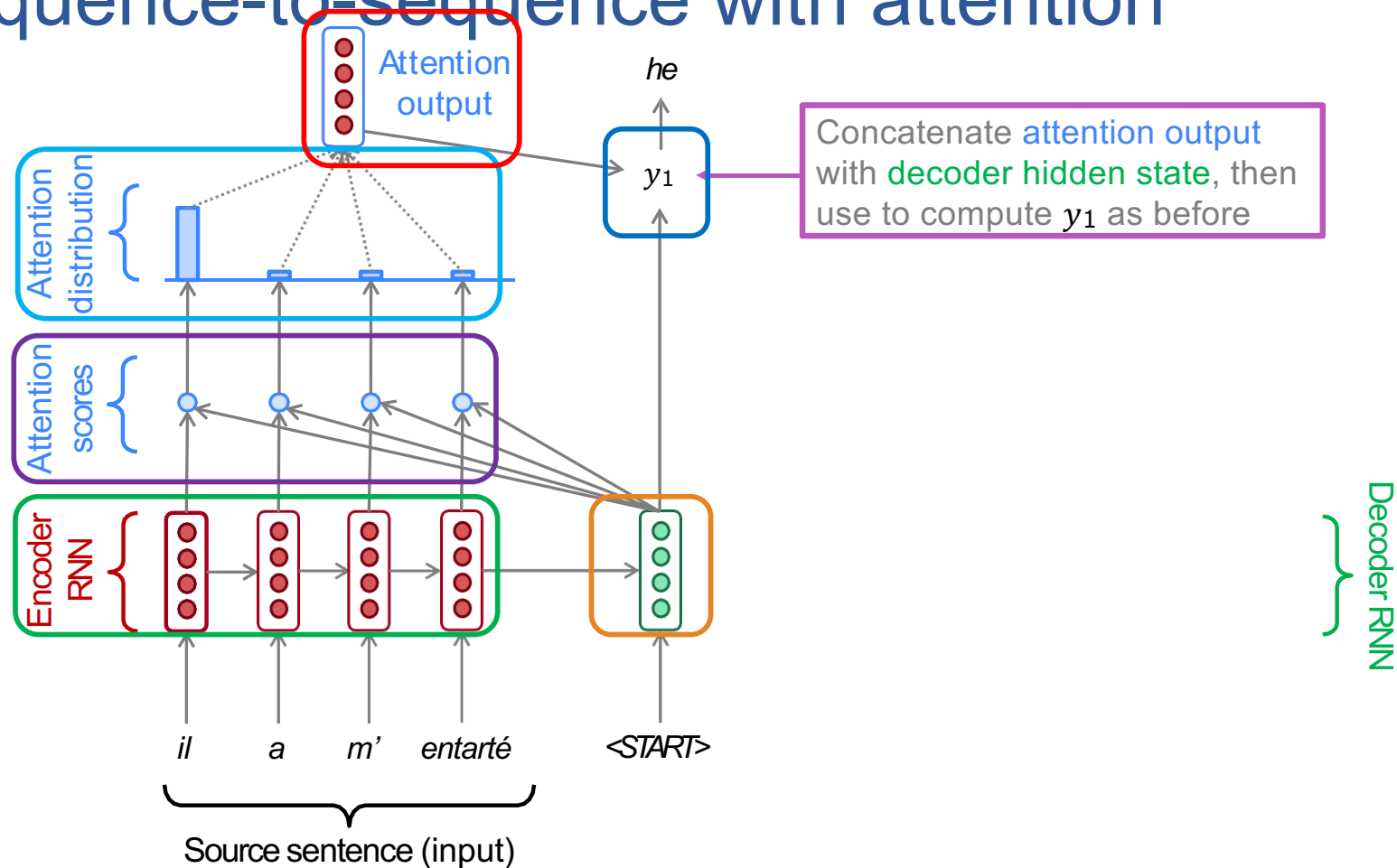
Sequence-to-sequence with attention



Sequence-to-sequence with attention



Sequence-to-sequence with attention



Attention in equations

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

$$e^t = [s_t^T h_1, \dots, s_t^T h_N] \in \mathbb{R}^N$$

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

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

- We use α^t to take a weighted sum of the encoder hidden states to get the **attention output** a_t

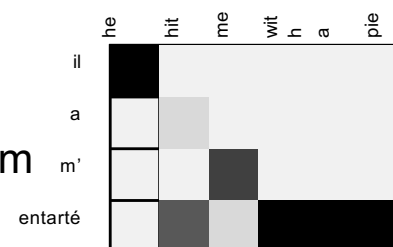
$$a_t = \sum_{i=1}^N \alpha_i^t h_i \in \mathbb{R}^h$$

- Finally, we **concatenate** the attention output a_t with the decoder hidden state s_t and proceed as in the non-attention seq2seq model

$$[a_t; s_t] \in \mathbb{R}^{2h}$$

Attention is great!

- Attention significantly **improves Neural Machine Trans. (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**).

Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models With Attention)

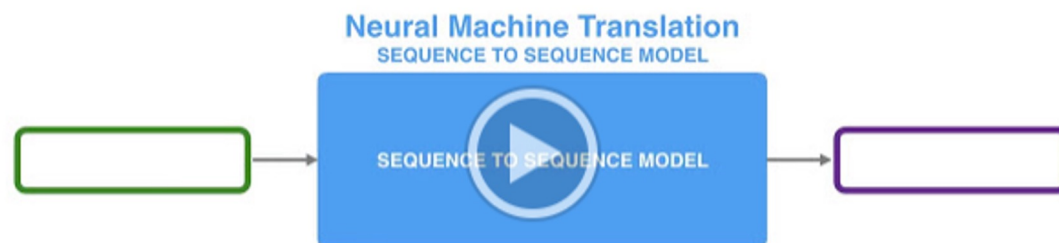
Translations: Chinese (Simplified), Japanese, Korean, Russian

Watch: MIT's [Deep Learning State of the Art](#) lecture referencing this post

May 25th update: New graphics (RNN animation, word embedding graph), color coding, elaborated on the final attention example.

Note: The animations below are videos. Touch or hover on them (if you're using a mouse) to get play controls so you can pause if needed.

Sequence-to-sequence models are deep learning models that have achieved a lot of success in tasks like machine translation, text summarization, and image captioning. Google Translate started [using](#) such a model in production in late 2016. These models are explained in the two pioneering papers ([Sutskever et al., 2014](#), [Cho et al., 2014](#)).



<https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/>

<https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/>

Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models With Attention)

Attention at time step 4

1. Prepare inputs

h_1

h_2

h_3

Encoder hidden states

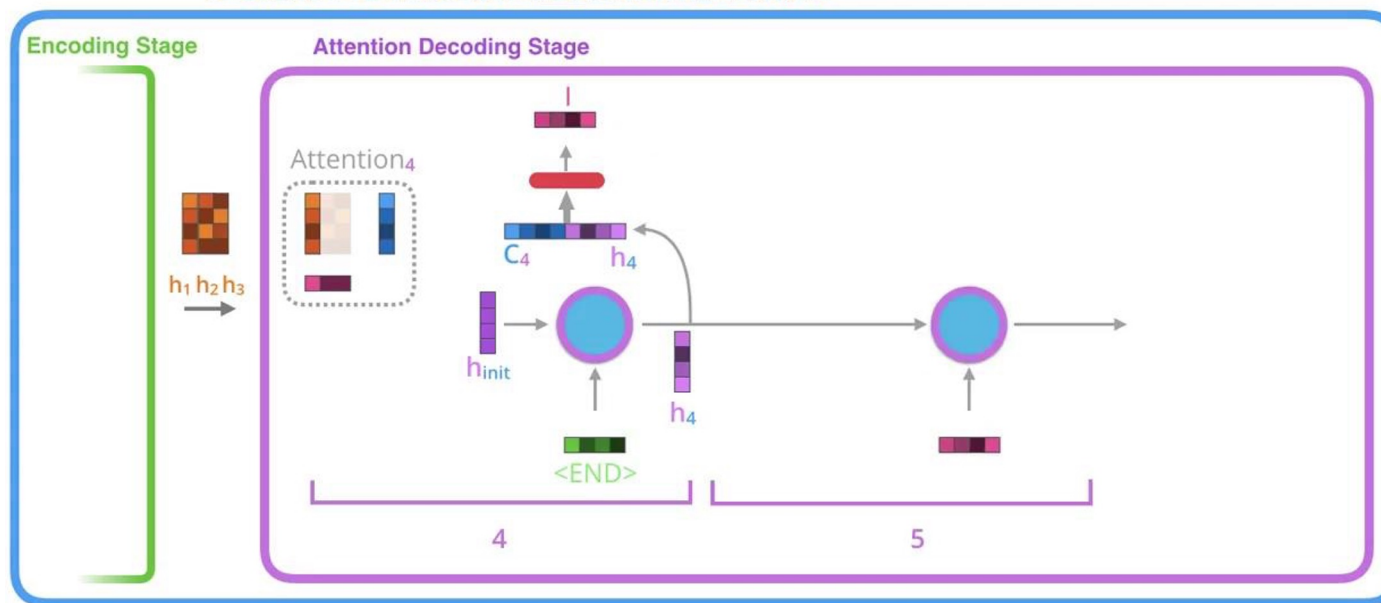
Decoder hidden state at time step 4

<https://falammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/>

Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models With Attention)

Neural Machine Translation SEQUENCE TO SEQUENCE MODEL WITH ATTENTION



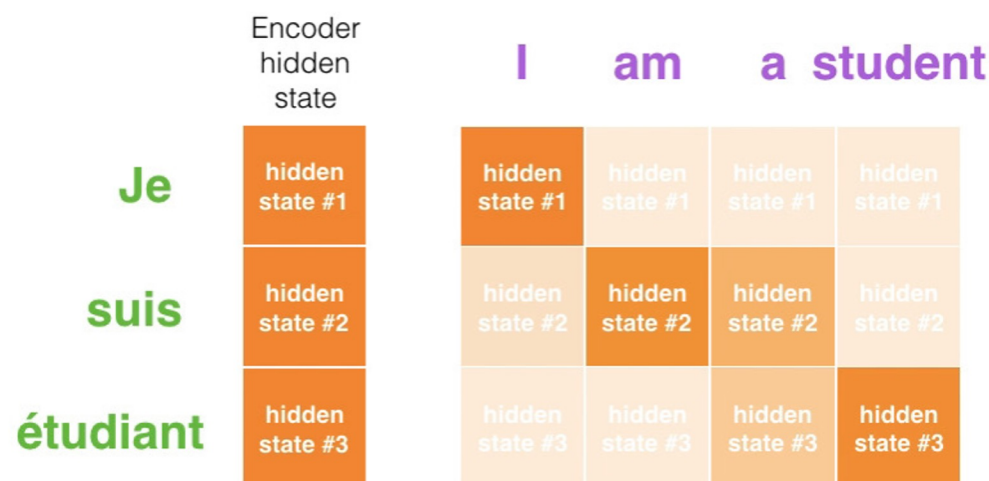
<https://alammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/>

Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models With Attention)

Translations: Chinese (Simplified), Japanese, Korean, Russian

Watch: MIT's [Deep Learning State of the Art](#) lecture referencing this post

May 25th update: New graphics (RNN animation, word embedding graph), color coding, elaborated on the final attention example.

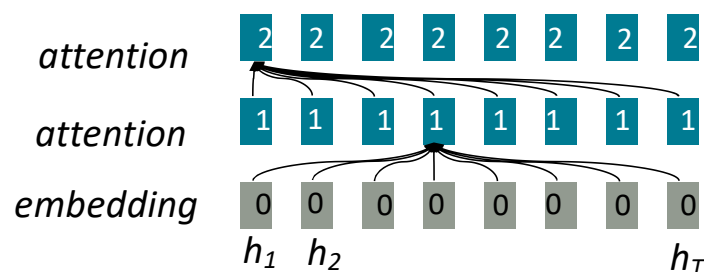


<https://falammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/>

Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

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

Self-Attention

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

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

$$e_{ij} = q_i^T k_j$$

Compute **key-query** affinities

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})}$$

Compute **attention weights** from affinities (softmax)

$$\text{output}_i = \sum_j \alpha_{ij} v_j$$

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**

$p_i \in \mathbb{R}^d$, for $i \in \{1, 2, \dots, T\}$ 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**.

$$v_i = v_i' + p_i$$

$$q_i = q_i' + p_i$$

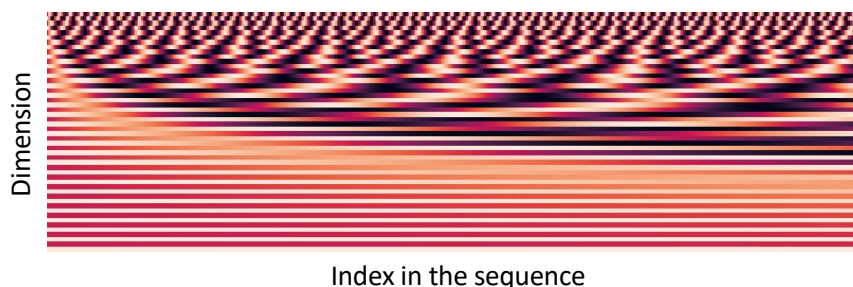
$$k_i = k_i' + p_i$$

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

- Doesn't have an inherent notion of order!
- No nonlinearities for deep learning! It's all just weighted averages



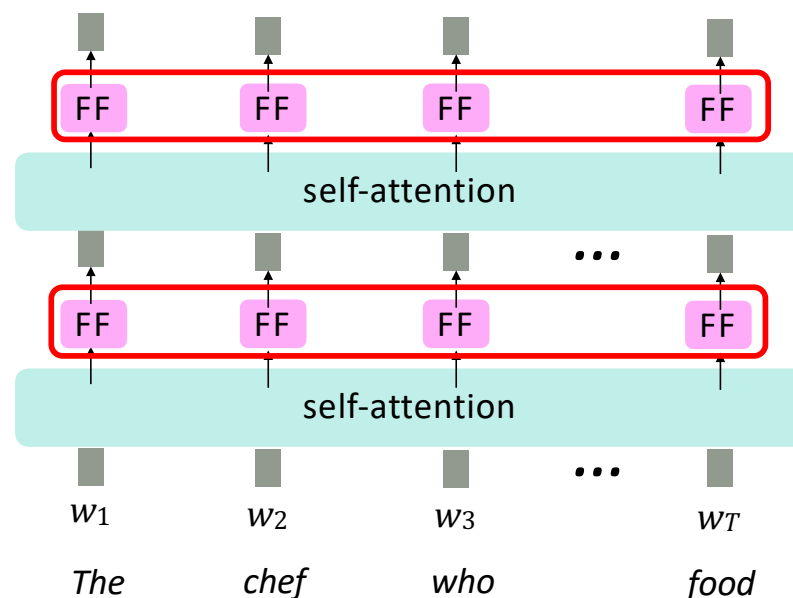
Solutions

- Add position representations to the inputs

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.

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



Intuition: the FF network processes the result of attention

Barriers and solutions for Self-Attention as a building block

Barriers

- Doesn't have an inherent notion of order!
- No nonlinearities for deep learning magic! It's all just weighted averages
- Need to ensure we don't "look at the future" when predicting a sequence
 - Like in machine translation
 - Or language modeling



Solutions

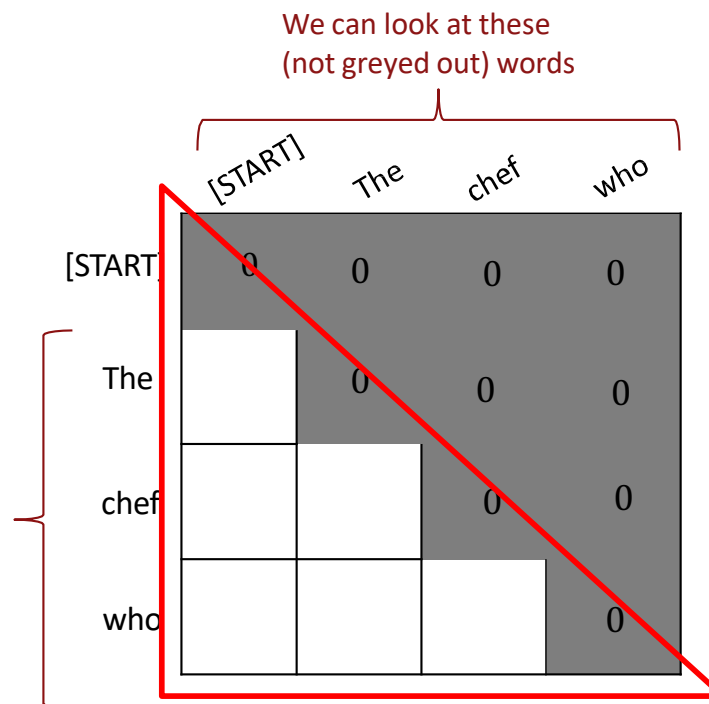
- Add position representations to the inputs
- Easy fix: apply the same feedforward network to each self-attention output.

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.

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

For encoding these words



Barriers and solutions for Self-Attention as a building block

Barriers

- Doesn't have an inherent notion of order!
- No nonlinearities for deep learning magic! It's all just weighted averages
- Need to ensure we don't "look at the future" when predicting a sequence
 - Like in machine translation
 - Or language modeling



Solutions

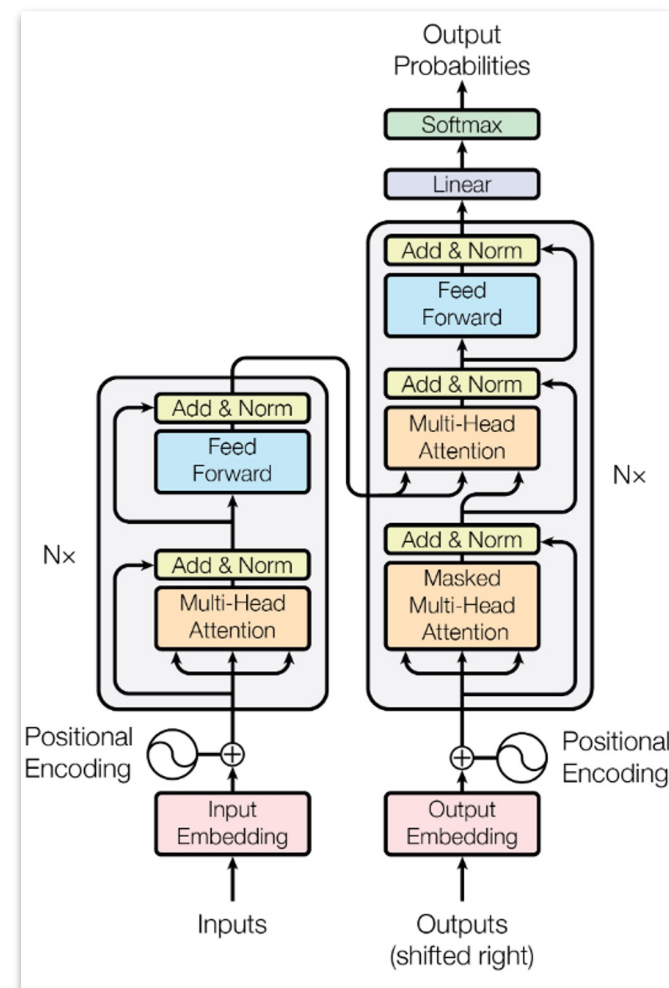
- Add position representations to the inputs
- Easy fix: apply the same feedforward network to each self-attention output.
- Mask out the future by artificially setting attention weights to 0!

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

- Transformer Architecture
 - Encoder & Decoder
 - Input & output embedding
 - Positional encoding
 - Self-attention
 - Multi-head attention
 - Masked multi-head attention
 - Residual connections
 - Layer Normalization
 - Feedforward



Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

<https://jalammar.github.io/illustrated-transformer/>



The Illustrated Transformer

Discussions: [Hacker News \(65 points, 4 comments\)](#), [Reddit r/MachineLearning \(29 points, 3 comments\)](#)

Translations: [Chinese \(Simplified\)](#), [French](#), [Japanese](#), [Korean](#), [Russian](#), [Spanish](#)

Watch: MIT's [Deep Learning State of the Art](#) lecture referencing this post

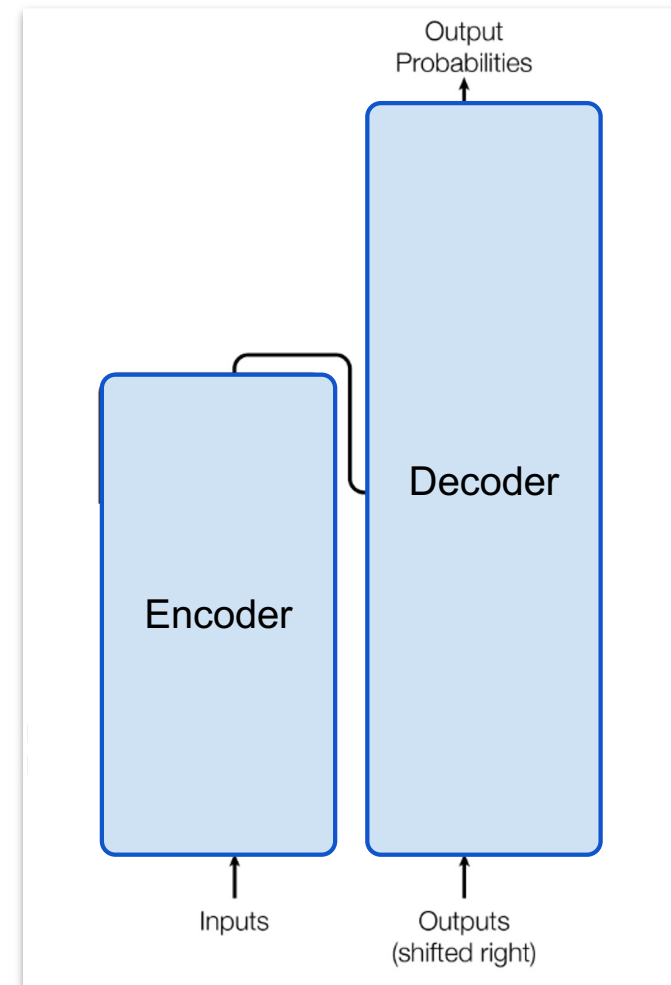
In the [previous post](#), we looked at [Attention](#) – a ubiquitous method in modern deep learning models. Attention is a concept that helped improve the performance of neural machine translation applications. In this post, we will look at **The Transformer** – a model that uses attention to boost the speed with which these models can be trained. The Transformers outperforms the Google Neural Machine Translation model in specific tasks. The biggest benefit, however, comes from how The Transformer lends itself to parallelization. It is in fact Google Cloud's recommendation to use The Transformer as a reference model to use their [Cloud TPU](#) offering. So let's try to break the model apart and look at how it functions.

The Transformer was proposed in the paper [Attention is All You Need](#). A TensorFlow implementation of it is available as a part of the [Tensor2Tensor](#) package. Harvard's NLP group created a [guide annotating the paper with PyTorch implementation](#). In this post, we will attempt to oversimplify things a bit and introduce the concepts one by one to hopefully make it easier to understand to people without in-depth knowledge of the subject matter.

2020 Update: I've created a "Narrated Transformer" video which is a gentler approach to the topic:

Transformer

- Transformer Architecture
 - **Encoder & Decoder**
 - Input & output embedding
 - Positional encoding
 - Self-attention
 - Multi-head attention
 - Masked multi-head attention
 - Residual connections
 - Layer Normalization
 - Feedforward

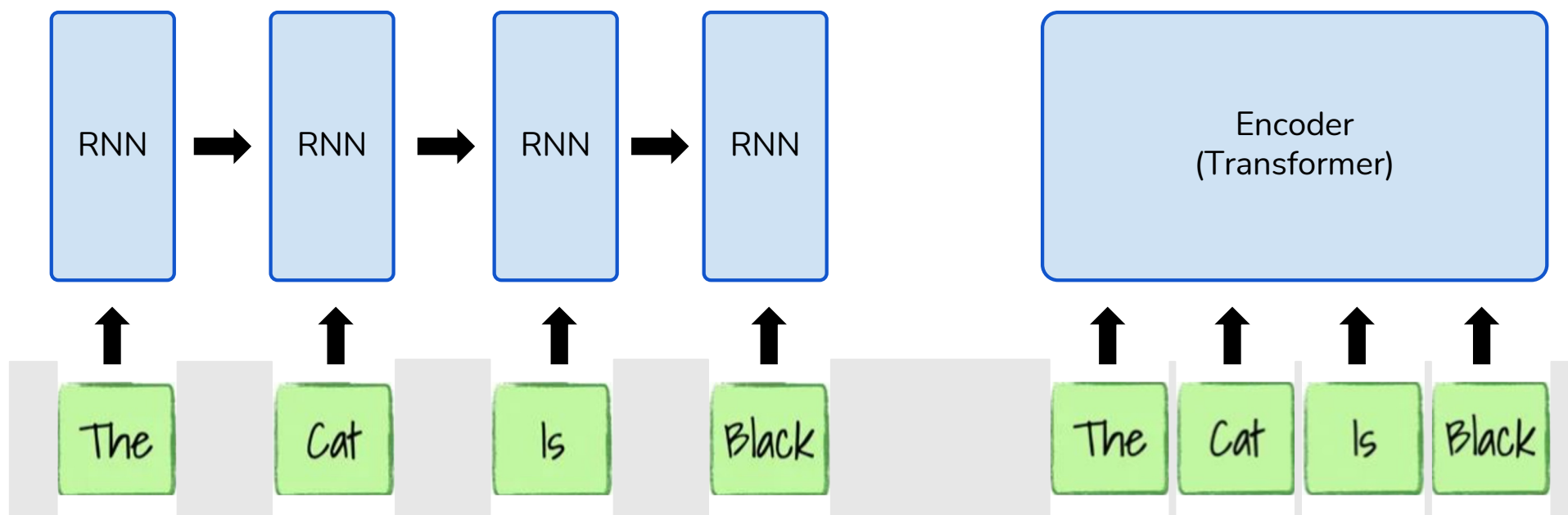


Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Transformer: Encoder & Decoder

- The model is primarily composed of two blocks:
 - The **encoder** receives an input and builds a representation of it (its features). This means that the model is optimized to acquire understanding from the input.
 - The **decoder** uses the encoder's representation (features) along with other inputs to generate a target sequence. This means that the model is optimized for generating outputs.

Transformers vs. RNNs

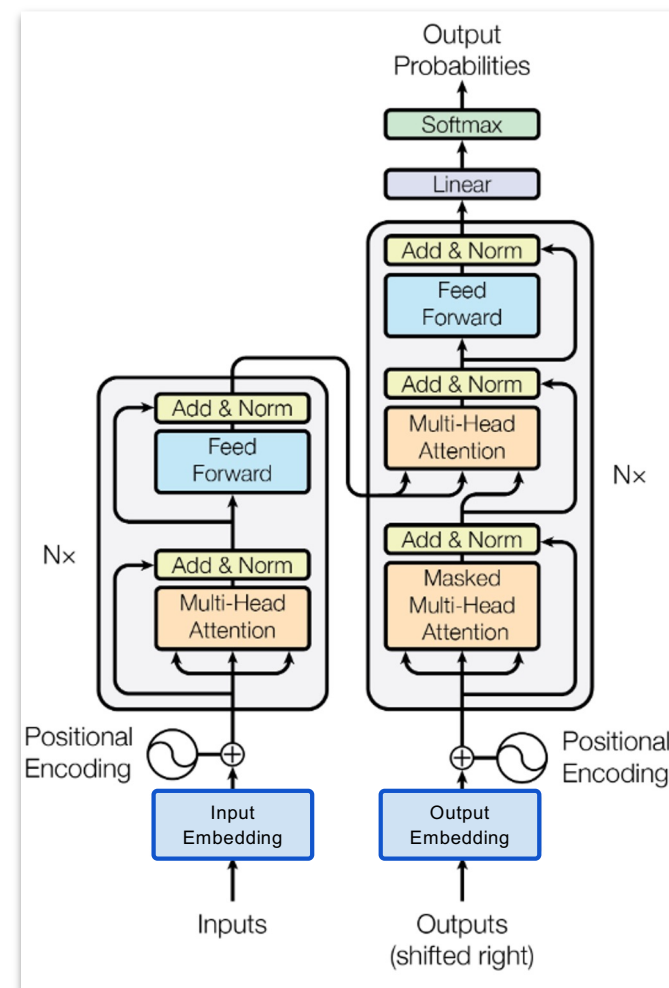


Transformer: Encoder & Decoder

- **Encoder-only models:** ALBERT, BERT, DistilBERT, ELECTRA, RoBERTa
- **Decoder-only models:** CTRL, GPT, GPT-2, GPT-3, GPT-4, Transformer XL.
- **Encoder-decoder models or sequence-to-sequence models:** BART, mBART, Marian, T5.

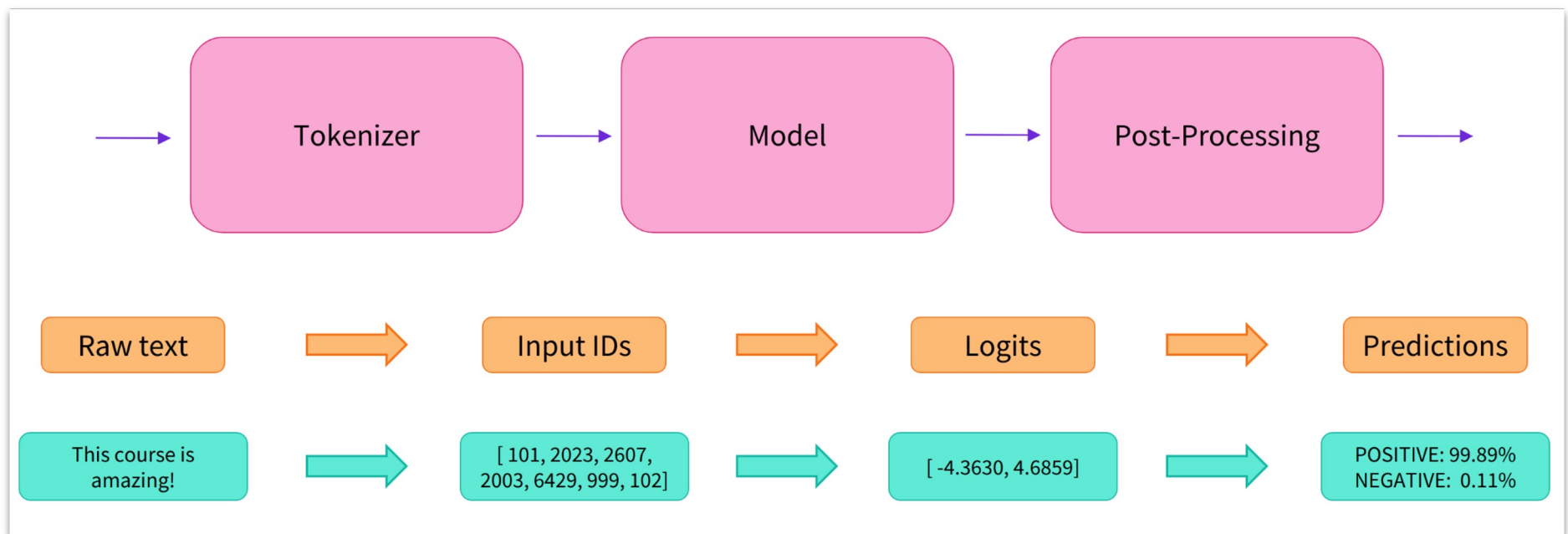
Transformer

- Transformer Architecture
 - Encoder & Decoder
 - Input & output embedding**
 - Positional encoding
 - Self-attention
 - Multi-head attention
 - Masked multi-head attention
 - Residual connections
 - Layer Normalization
 - Feedforward



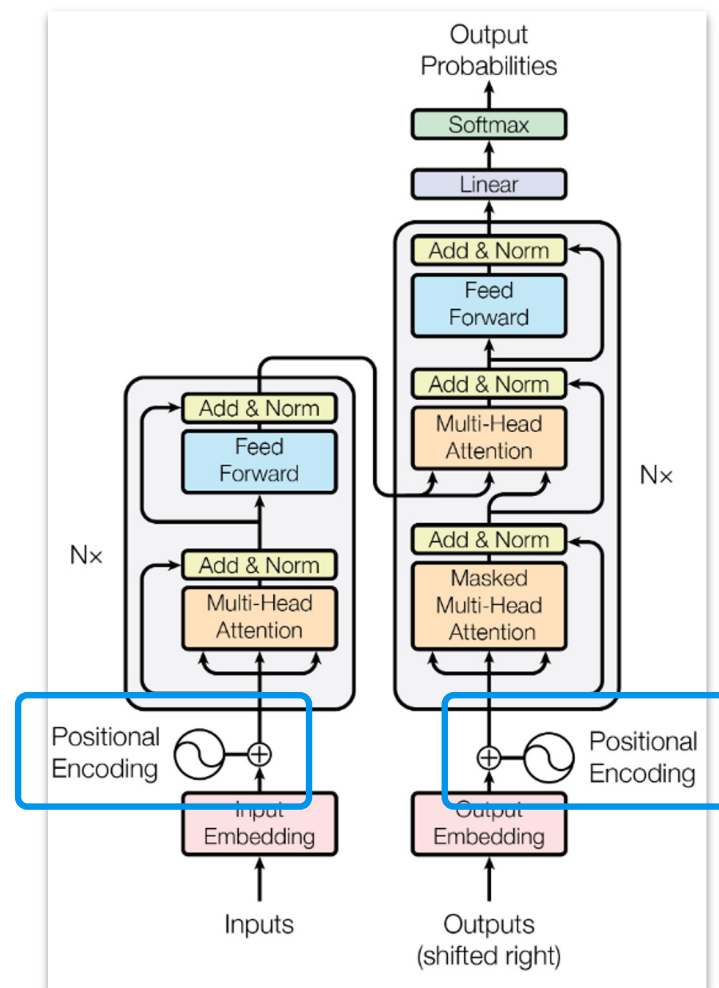
Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Transformer: Input & output embedding



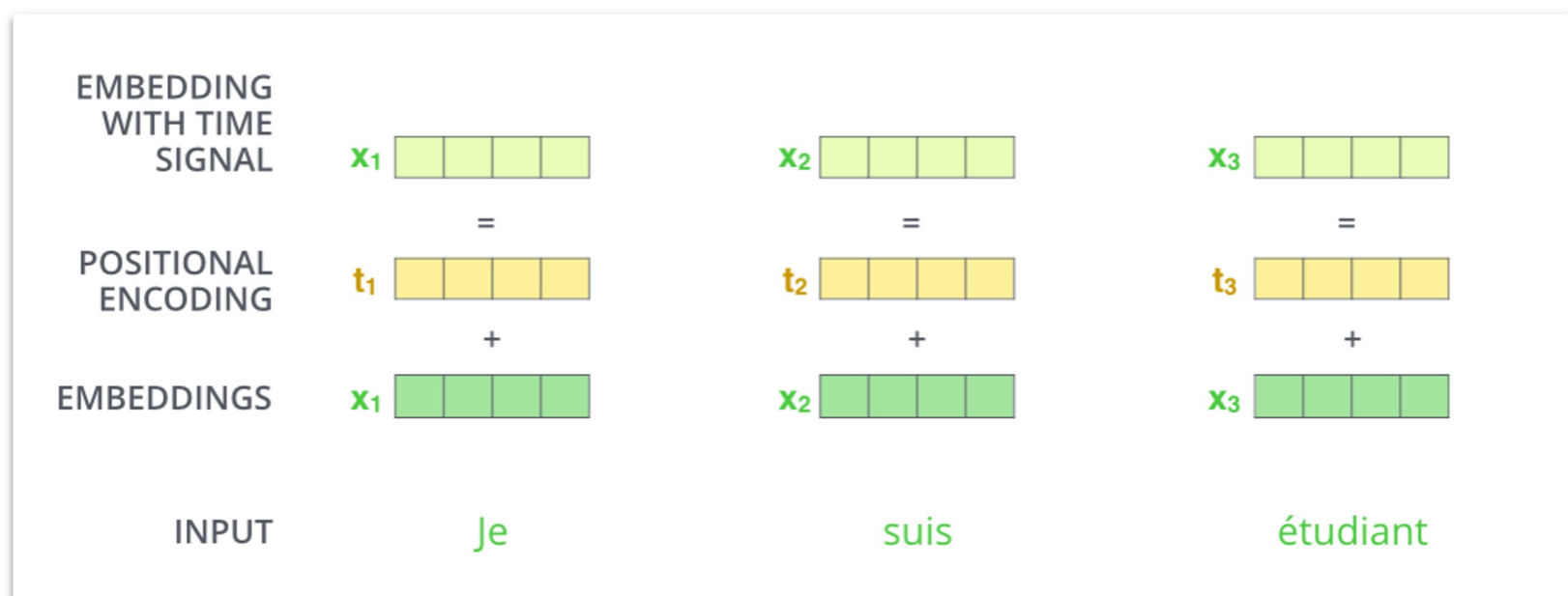
Transformer

- Transformer Architecture
 - Encoder & Decoder
 - Input & output embedding
 - Positional encoding**
 - Self-attention
 - Multi-head attention
 - Masked multi-head attention
 - Residual connections
 - Layer Normalization
 - Feedforward



Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Transformer: Positional Encoding



Transformer: Positional Encoding

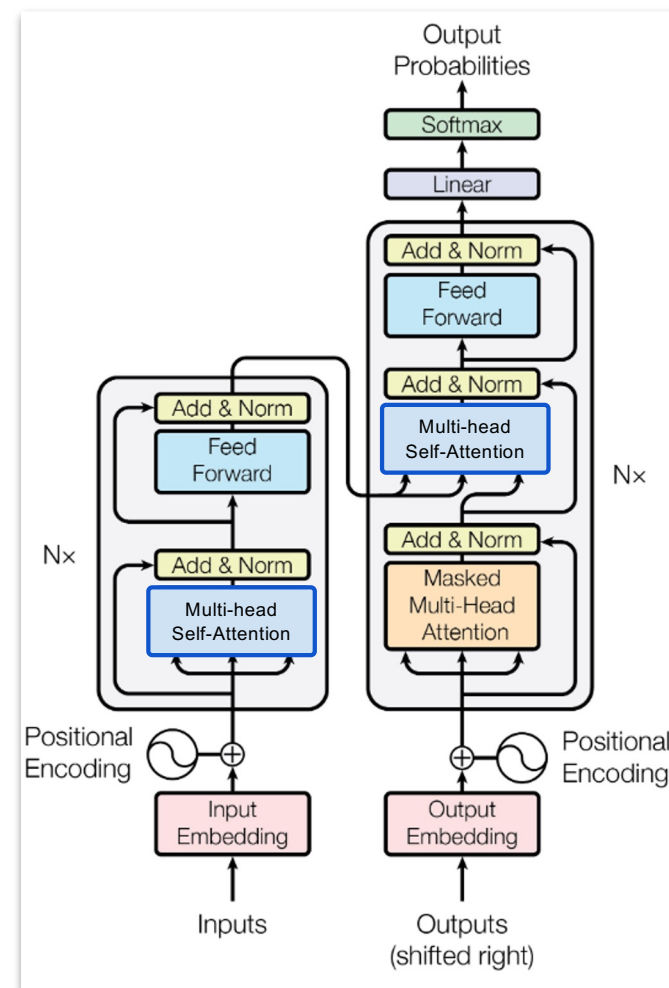
$$PE_{(pos, 2i)} = \sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos, 2i+1)} = \cos(pos/10000^{2i/d_{\text{model}}})$$



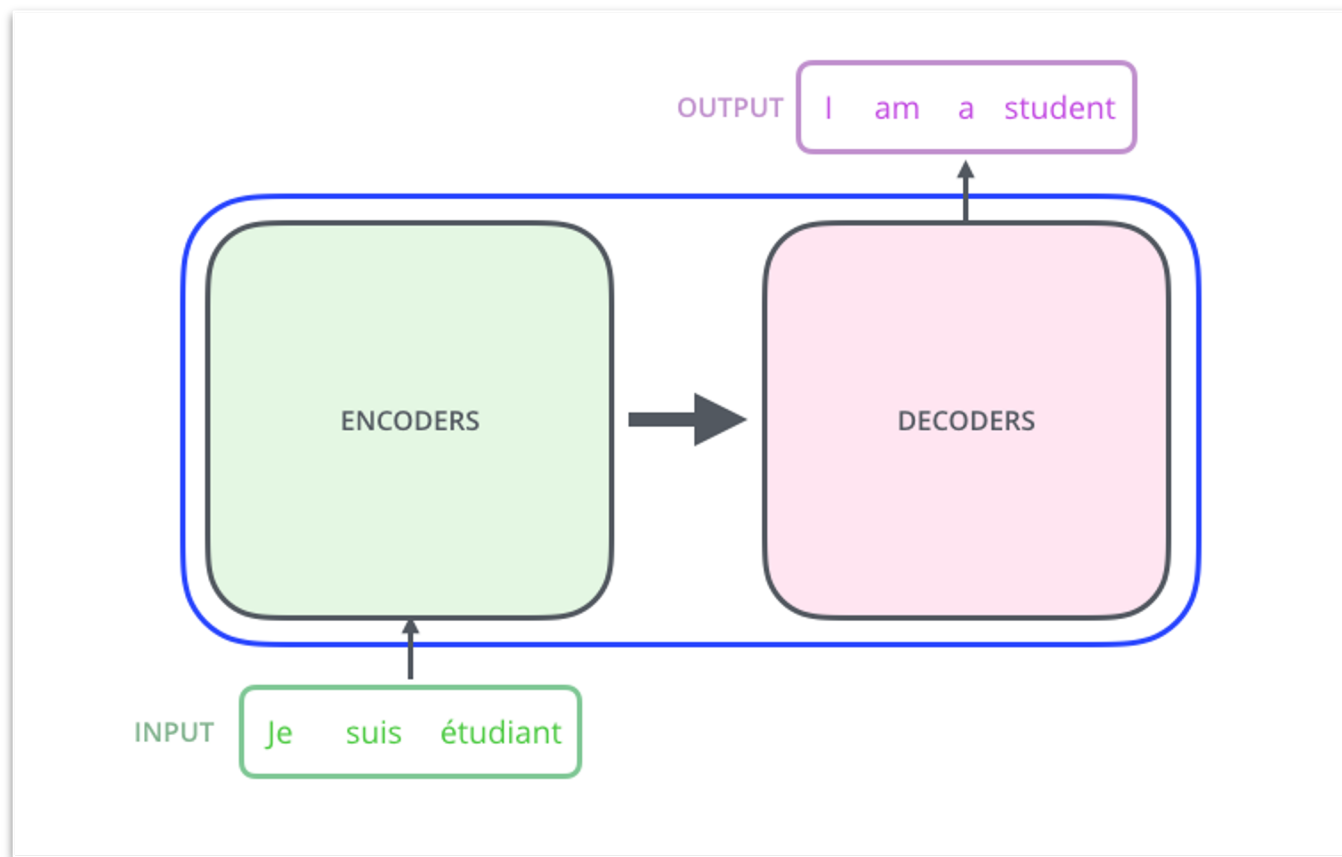
Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Transformer

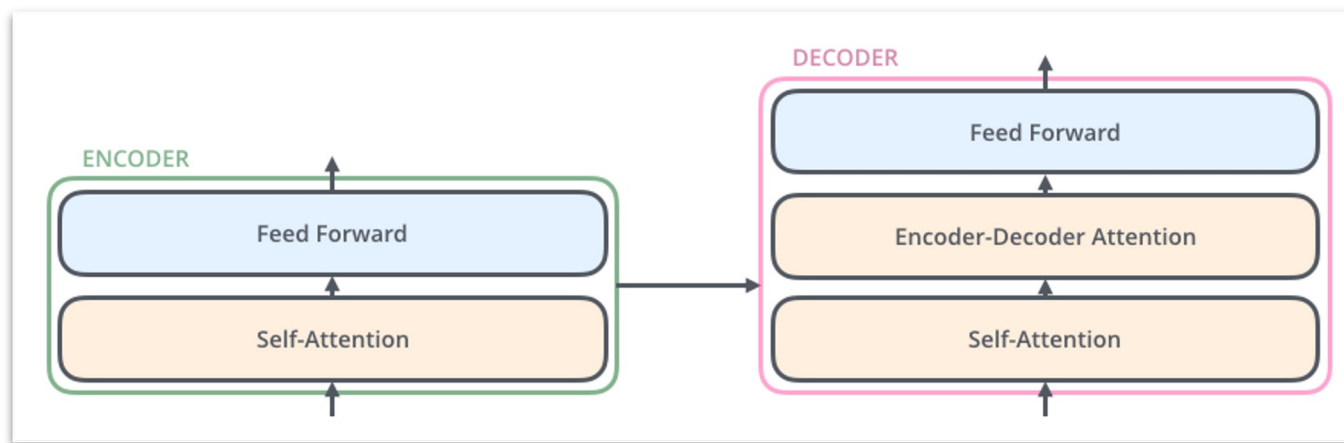
- Transformer Architecture
 - Encoder & Decoder
 - Input & output embedding
 - Positional encoding
 - Self-attention
 - Multi-head attention
 - Masked multi-head attention
 - Residual connections
 - Layer Normalization
 - Feedforward



Slide Credit: [Prof. Sandra Avila](#) - UNICAMP



Slide Credit: [Prof. Sandra Avila](#) - UNICAMP



Slide Credit: [Prof. Sandra Avila](https://ialammar.github.io/illustrated-transformer) - UNICAMP

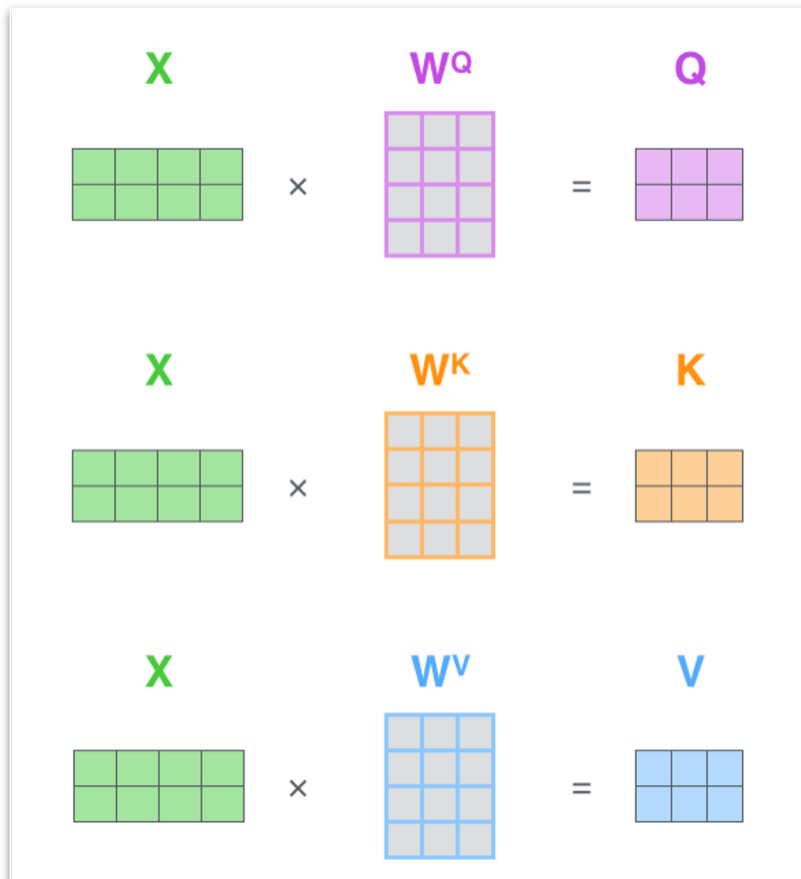
The Transformer Encoder: Dot-Product Attention

- 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 , values have d_v

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

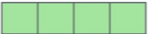
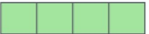










The Transformer Encoder: Key-Query-Value Attention

- 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, \dots, x_T be **input vectors** to the Transformer encoder; $x_i \in \mathbb{R}^{d_1}$
- Then **keys**, **queries**, **values** are:
 - $k_i = x_i^T W^K$, where $W^K \in \mathbb{R}^{d_1 \times d_2}$ is the key matrix.
 - $q_i = x_i^T W^Q$, where $W^Q \in \mathbb{R}^{d_1 \times d_2}$ is the query matrix.
 - $v_i = x_i^T W^V$, where $W^V \in \mathbb{R}^{d_1 \times d_2}$ is the value matrix.
- These matrices allow *different aspects* of the x vectors to be used/emphasized in each of the three roles.




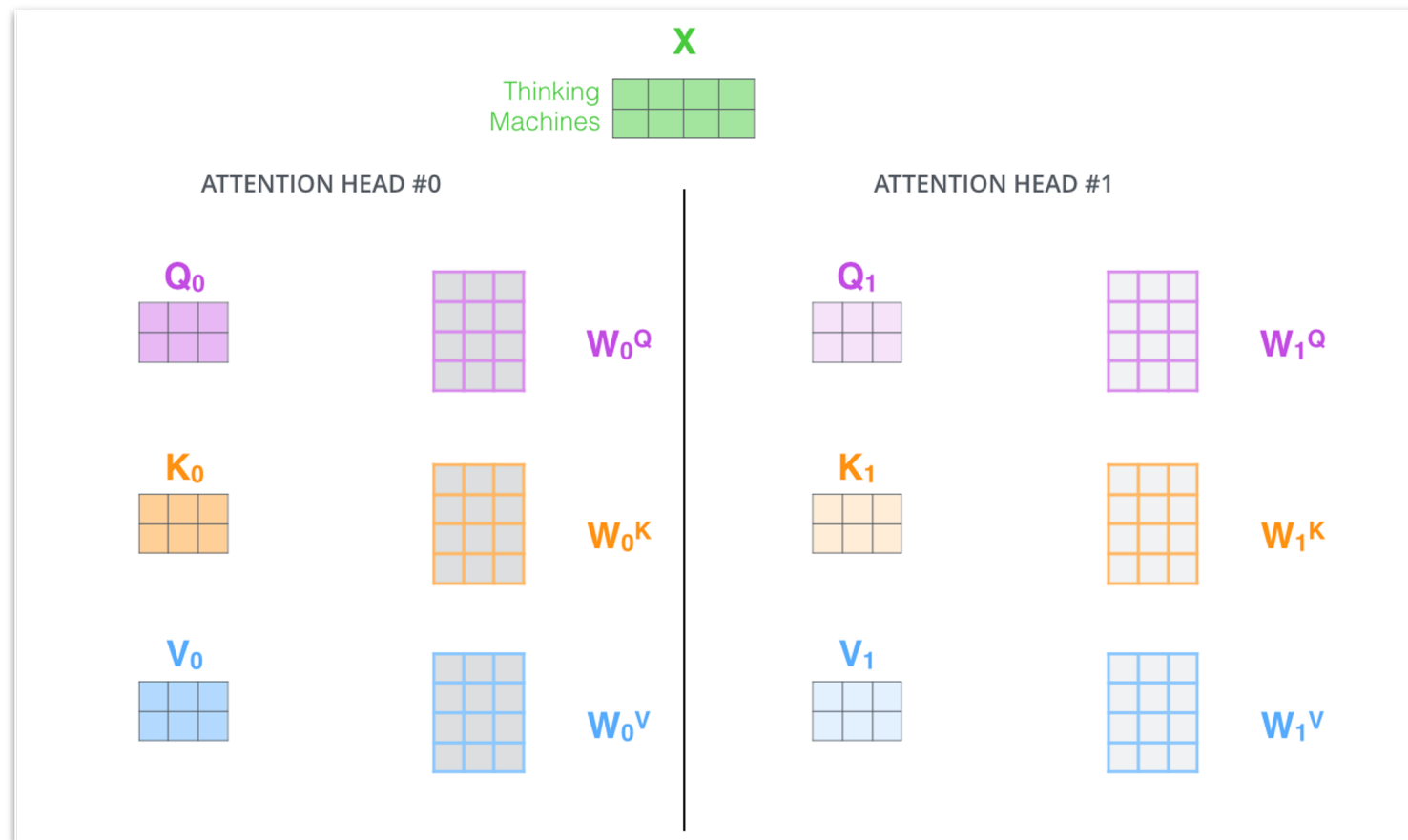
$$\text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) \times V = Z$$

The diagram shows the calculation of the Z matrix. The Q matrix (2x4 purple grid) is multiplied by the transpose of the K matrix (2x4 orange grid, labeled K^T). The result is divided by $\sqrt{d_k}$ and passed through a softmax function. The result is then multiplied by the V matrix (2x4 blue grid) to produce the final Z matrix (2x4 pink grid).

Input	Thinking	Machines
Embedding	x_1 	x_2 
Queries	q_1 	q_2 
Keys	k_1 	k_2 
Values	v_1 	v_2 
Score	$q_1 \cdot k_1 = 112$	$q_2 \cdot k_2 = 96$
Divide by 8 ($\sqrt{d_k}$)	14	12
Softmax	0.88	0.12
Softmax X Value	v_1 	v_2 
Sum	z_1 	z_2 

$$\text{softmax} \left(\frac{Q \times K^T}{\sqrt{d_k}} \right) V$$

= Z 

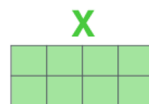


Slide Credit: [Prof. Sandra Avila](https://ialammar.github.io/illustrated-transformer) - UNICAMP

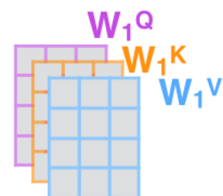
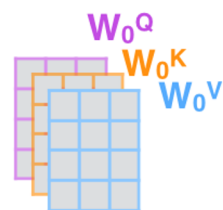
1) This is our
input sentence*

Thinking
Machines

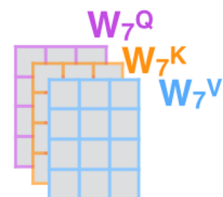
2) We embed
each word*



3) Split into 8 heads.
We multiply X or
 R with weight matrices



...



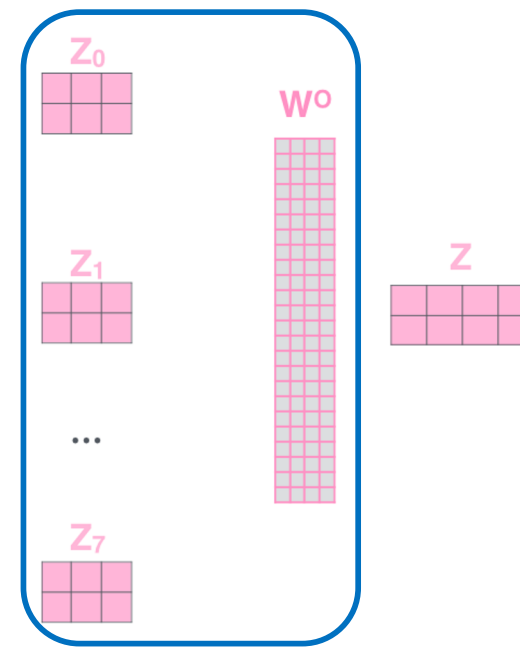
4) Calculate attention
using the resulting
 $Q/K/V$ matrices



...



5) Concatenate the resulting Z matrices,
then multiply with weight matrix W^O to
produce the output of the layer

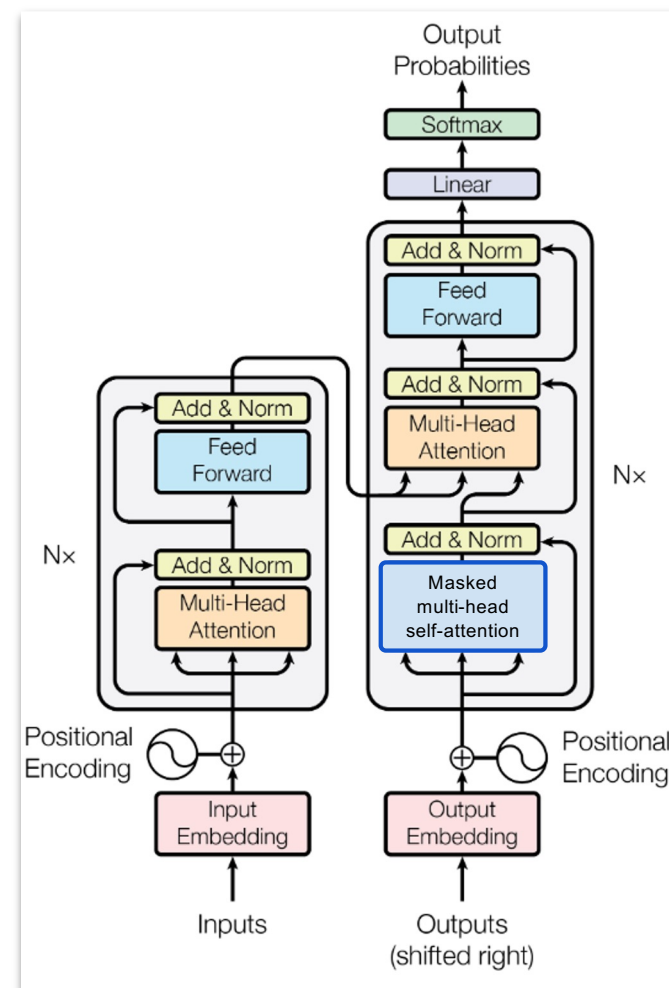


Linear Layer

Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Transformer

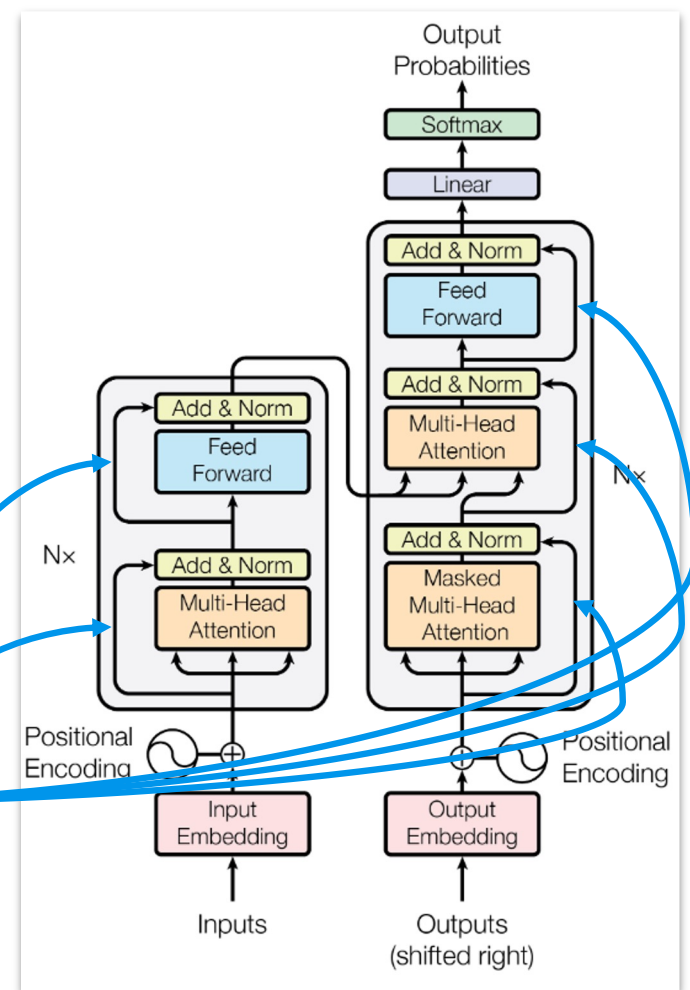
- Transformer Architecture
 - Encoder & Decoder
 - Input & output embedding
 - Positional encoding
 - Self-attention
 - Multi-head attention
 - Masked multi-head attention**
 - Residual connections
 - Layer Normalization
 - Feedforward



Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Transformer

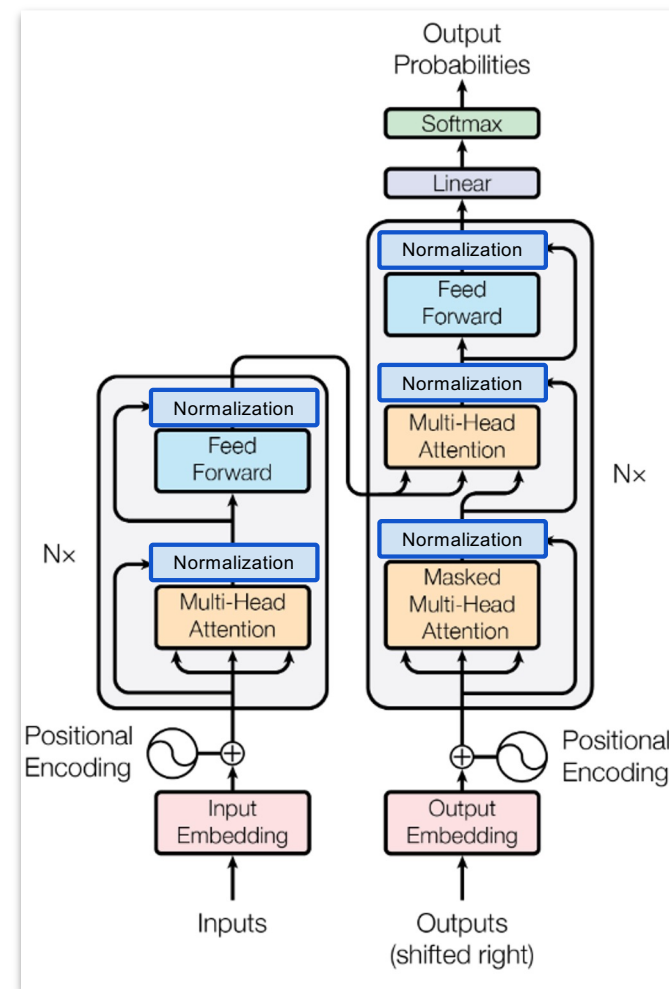
- Transformer Architecture
 - Encoder & Decoder
 - Input & output embedding
 - Positional encoding
 - Self-attention
 - Multi-head attention
 - Masked multi-head attention
 - **Residual connections**
 - Layer Normalization
 - Feedforward



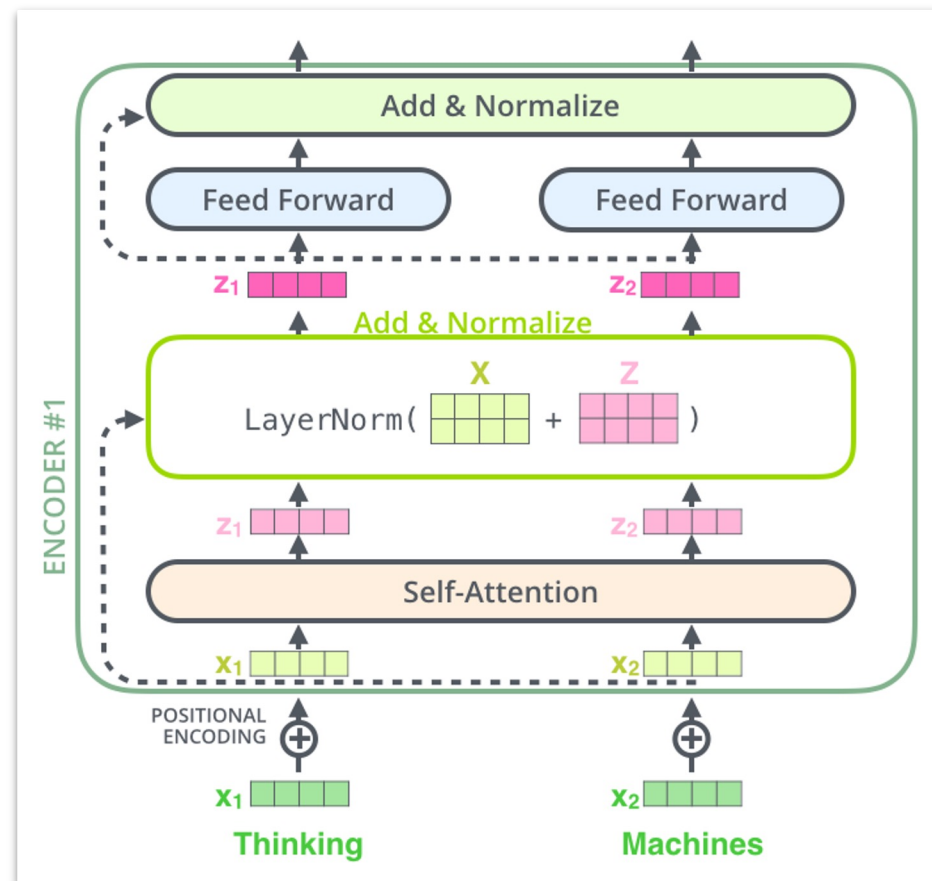
Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Transformer

- Transformer Architecture
 - Encoder & Decoder
 - Input & output embedding
 - Positional encoding
 - Self-attention
 - Multi-head attention
 - Masked multi-head attention
 - Residual connections
 - **Layer Normalization**
 - Feedforward

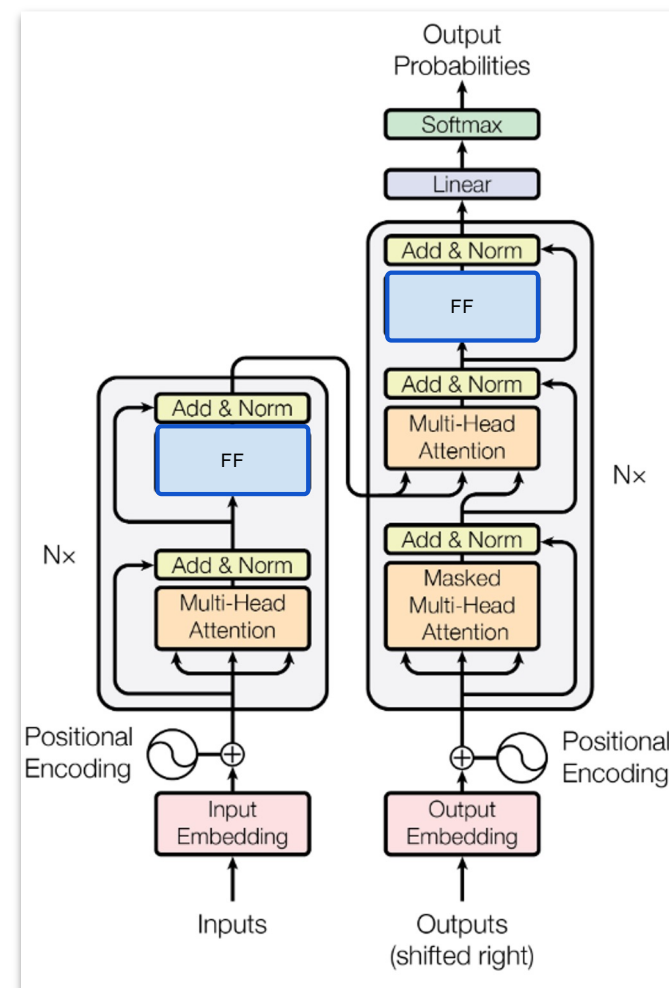


Slide Credit: [Prof. Sandra Avila](#) - UNICAMP



Transformer

- Transformer Architecture
 - Encoder & Decoder
 - Input & output embedding
 - Positional encoding
 - Self-attention
 - Multi-head attention
 - Masked multi-head attention
 - Residual connections
 - Layer Normalization
 - Feedforward**



Slide Credit: [Prof. Sandra Avila](#) - UNICAMP

Lab 12b: Transformers

Duration: 20 min



✕

Join at:
ahaslides.com/
52RQQ

To join, go to: ahaslides.com/52RQQ ✕

AhaSlides

Please, from Lab 12: Transformers [Section 6], submit your predicted rating.

The yelp review is → This is by far my favorite Panera location in the Pittsburgh area. Friendly, plenty of room to sit, and good quality food & coffee. Panera is a great place to hang out and read the news – they even have free WiFi! Try their toasted sandwiches, especially the chicken bacon dijon.

The Predicted Rating is → 3 and the Actual Rating was → 4

✓ Slide 1 selected for PowerPoint

☰

🔍

📁 Group

🗣️

👤 3

👥 1/100

✓

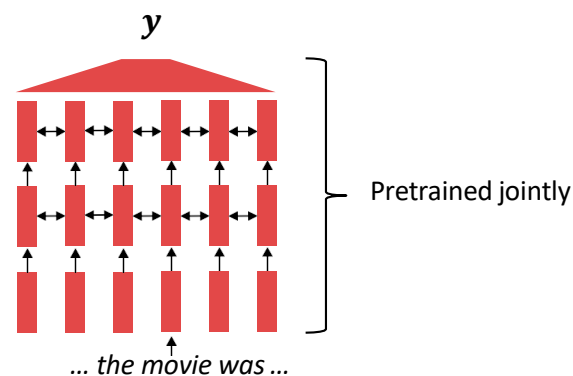
^

Get Feedback

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]

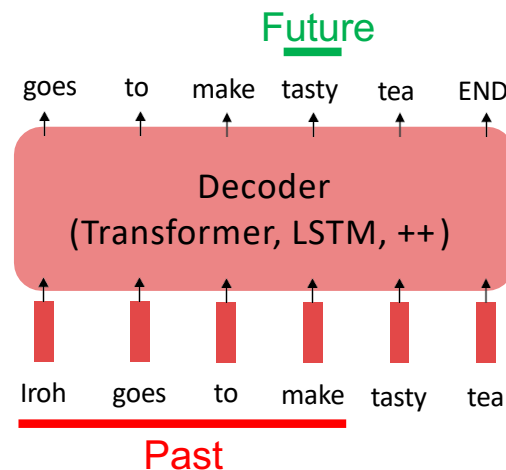
Pretraining through language modeling [\[Dai and Le, 2015\]](#)

Recall the **language modeling** task:

- Model $p_{\theta}(w_t | w_{1:t-1})$, 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.



Lab 12a

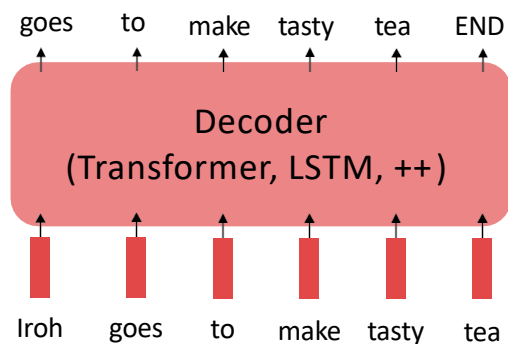


The Pretraining / Finetuning Paradigm

Pretraining can improve NLP applications by serving as parameter initialization.

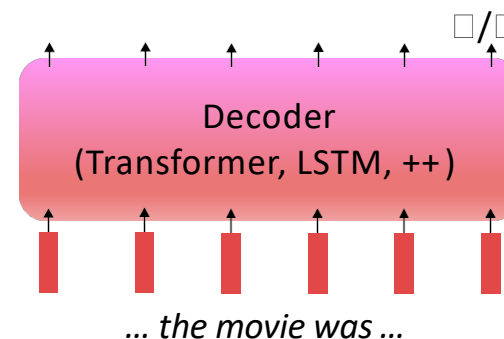
Step 1: Pretrain (on language modeling)

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”

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.

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..."
```

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>

Vicente Ordoñez

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

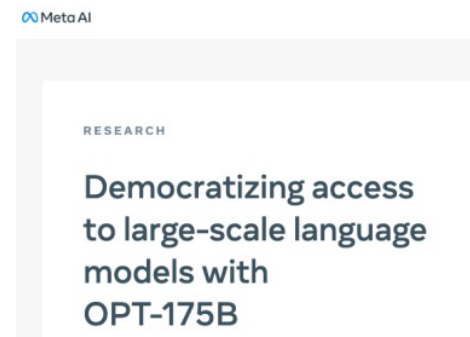
Output:

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>

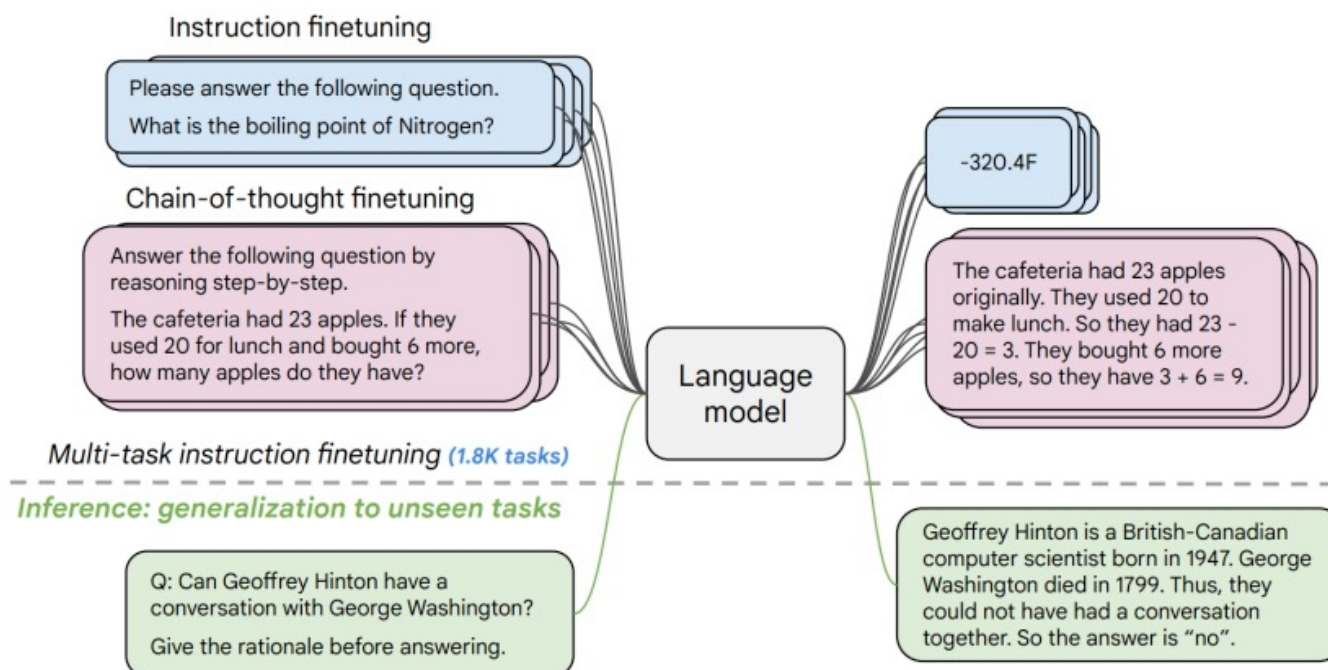


https://huggingface.co/docs/transformers/model_doc/opt

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)



Lab 12a



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)

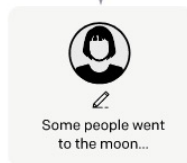
Step 1

**Collect demonstration data,
and train a supervised policy.**

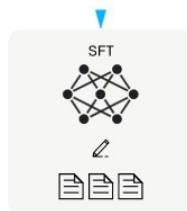
A prompt is
sampled from our
prompt dataset.



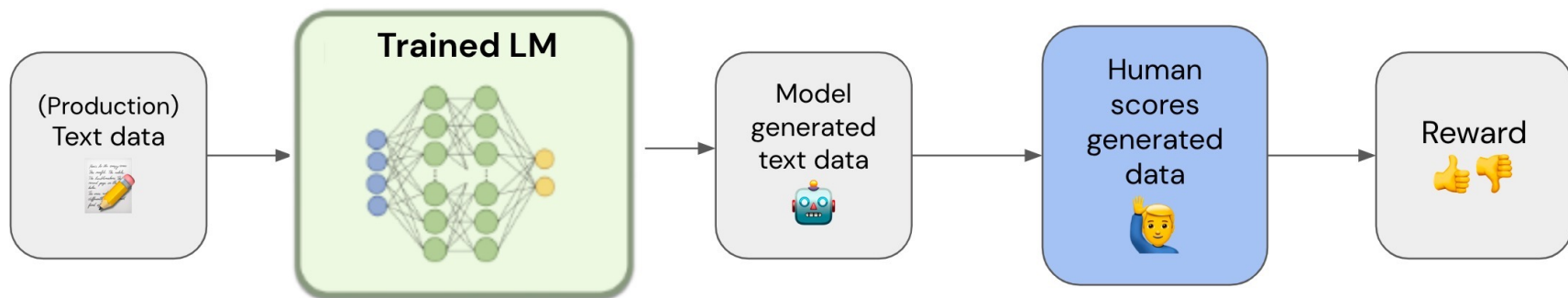
A labeler
demonstrates the
desired output
behavior.



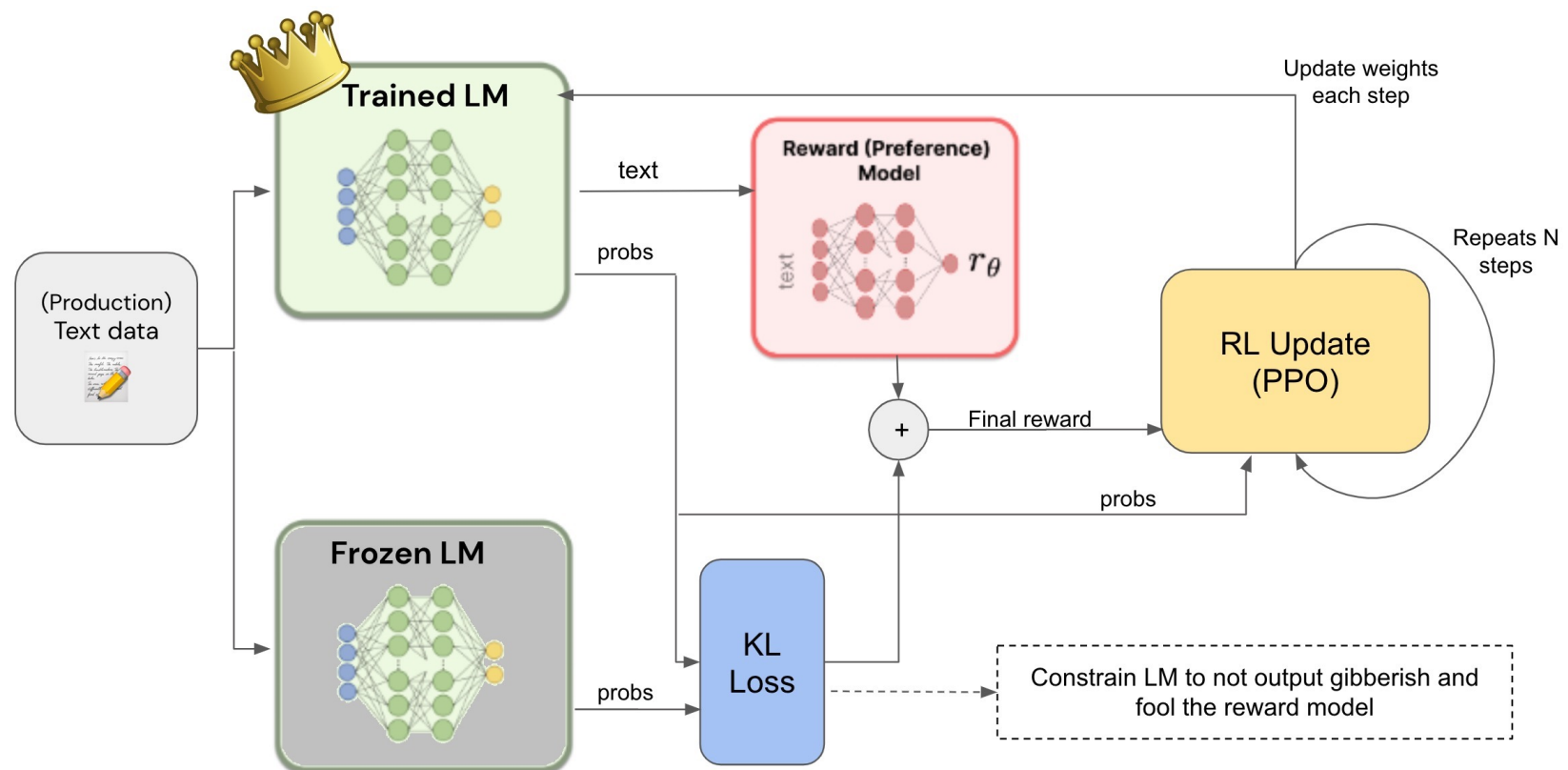
This data is used
to fine-tune GPT-3
with supervised
learning.



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



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



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!

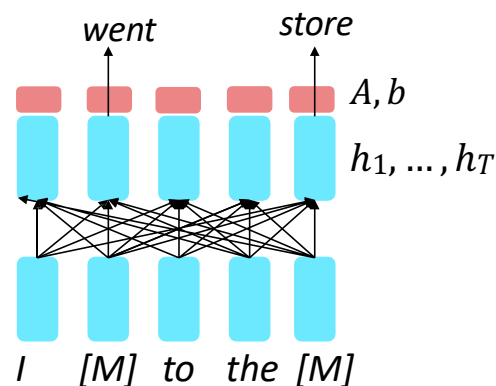
Idea: replace some fraction of words in the input with a special [MASK] token; predict these words.

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

$$y_i \sim Aw_i + b$$

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**.

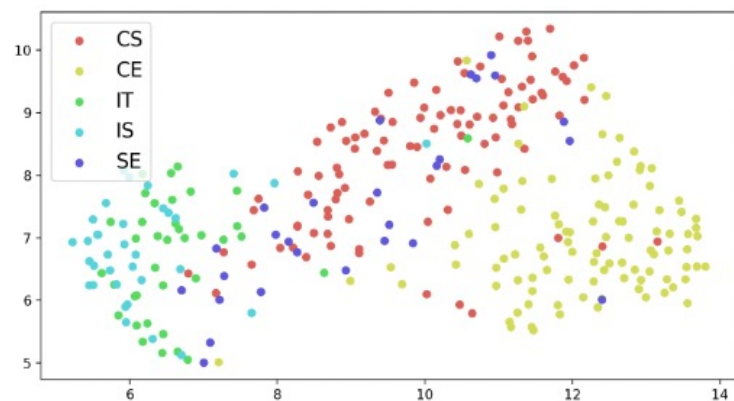
Example: **BERT**: Bidirectional Encoder Representations from Transformers



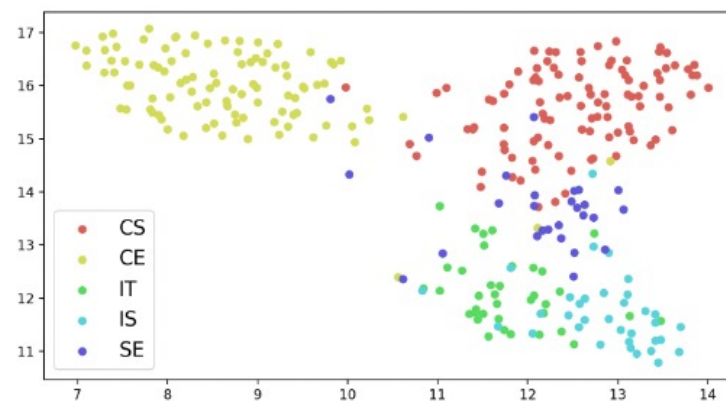
[Devlin et al., 2018]

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.



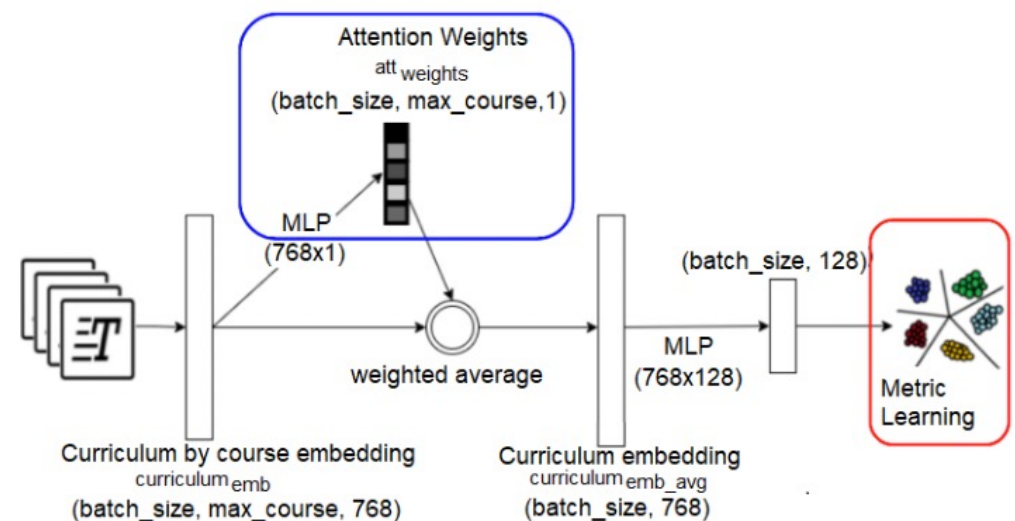
(a) *Bert*



(b) *Bert_{met+att}*

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.

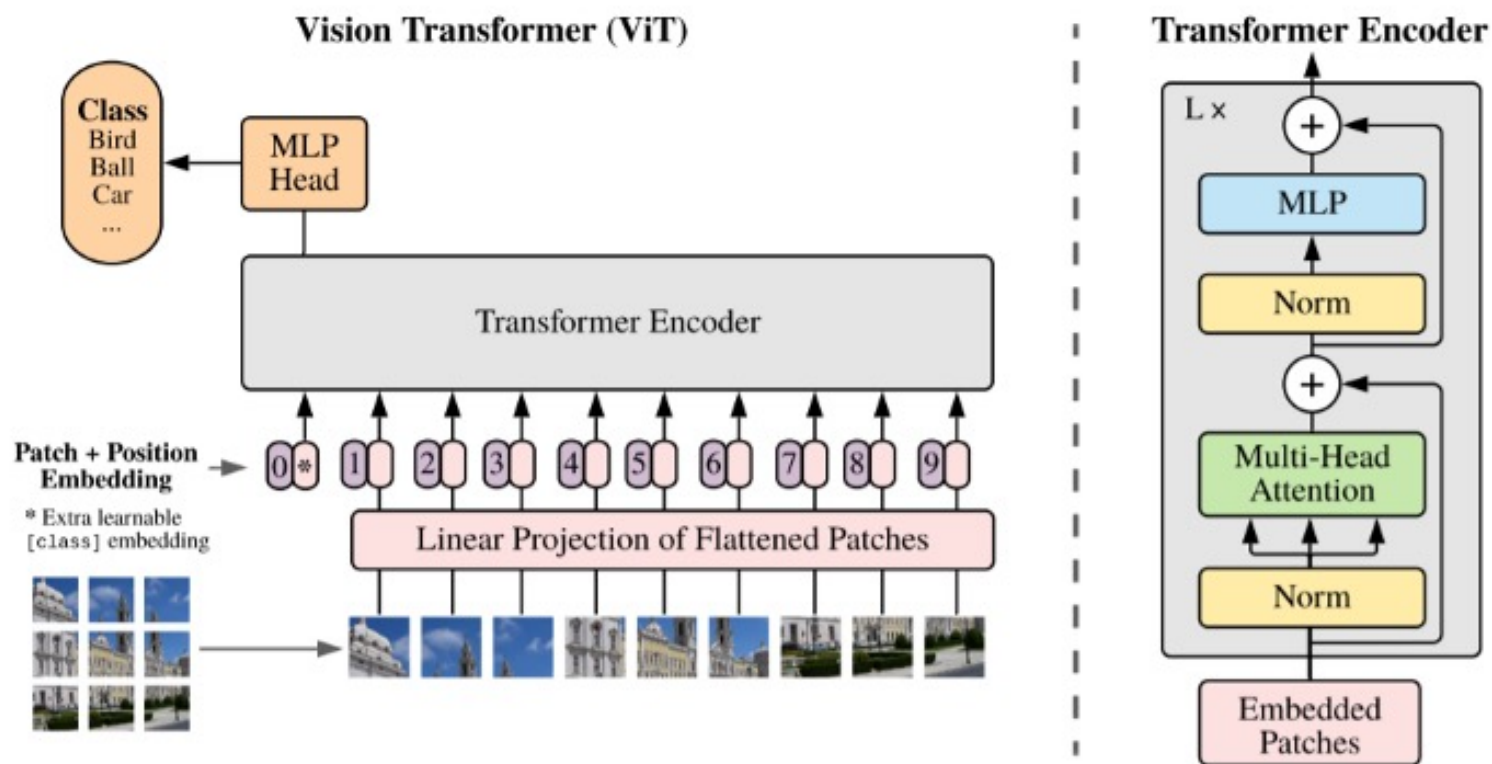


Capturing meaning via context: What kinds of things does pretraining learn?

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 Fibonacci sequence]
- Models also learn – and can exacerbate racism, sexism, all manner of bad biases.

Transformers in vision



Cross-modal transformers

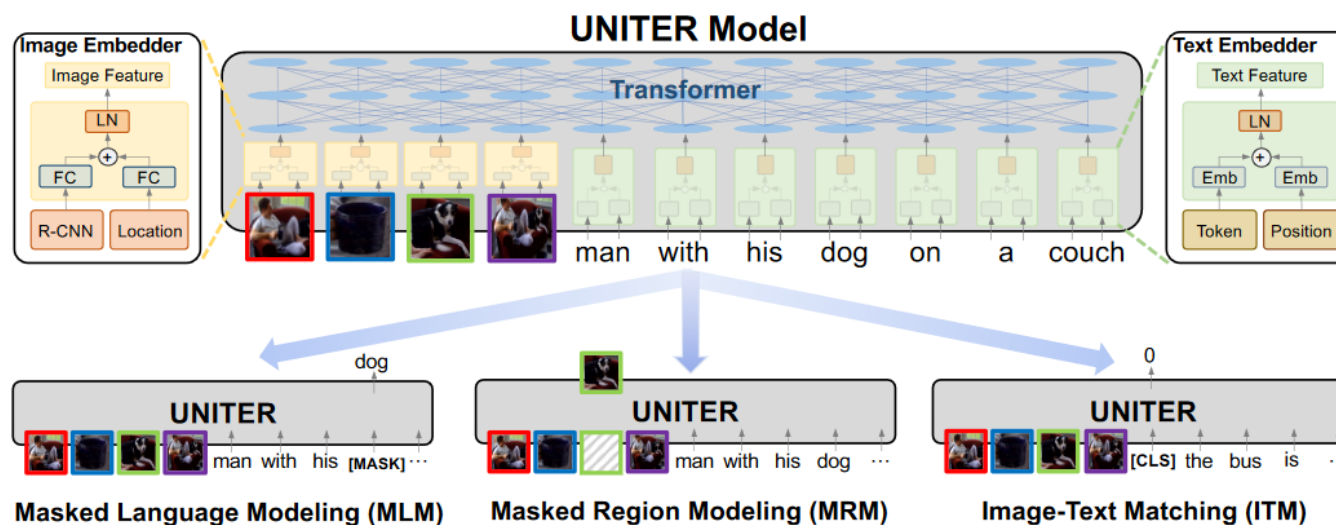
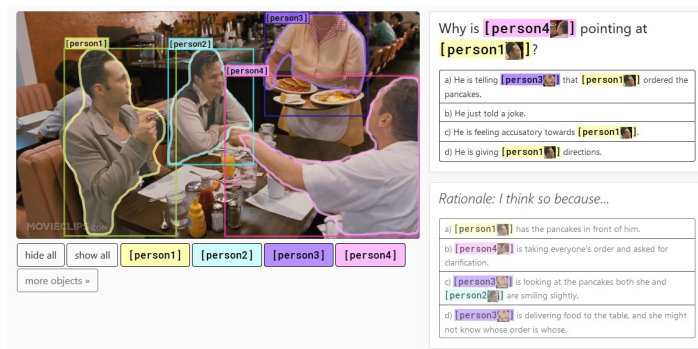


Figure 1: Overview of the proposed UNITER model (best viewed in color), consisting of an Image Embedder, a Text Embedder and a multi-layer self-attention Transformer, learned through three pre-training tasks.

Visual Commonsense Reasoning leaderboard



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

<https://visualcommonsense.com/leaderboard/>

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?



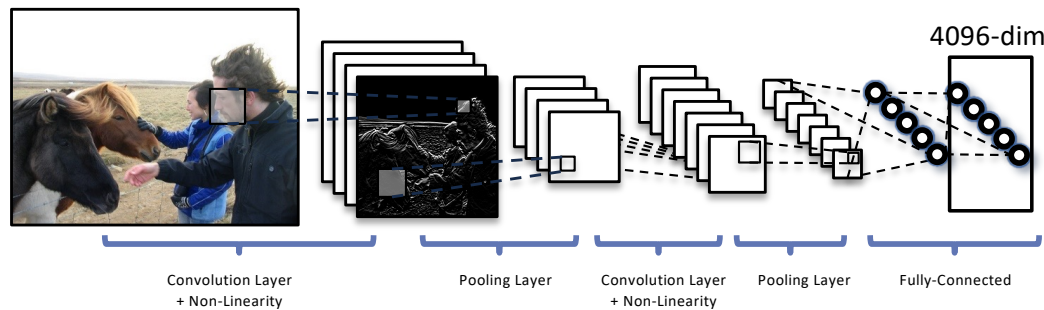
Is this person expecting company?
What is just under the tree?



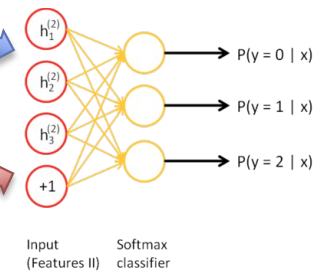
Does it appear to be rainy?
Does this person have 20/20 vision?

Visual Question Answering (VQA)

Image Embedding

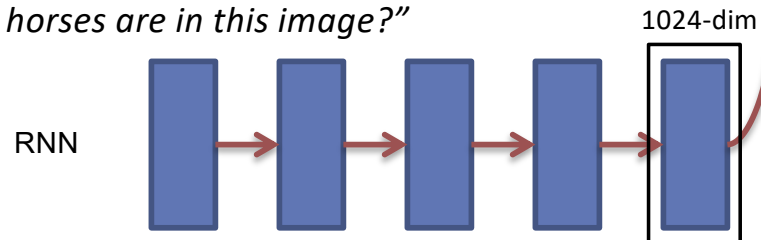


Neural Network
Softmax
over top K answers



Question Embedding

"How many horses are in this image?"



Extra

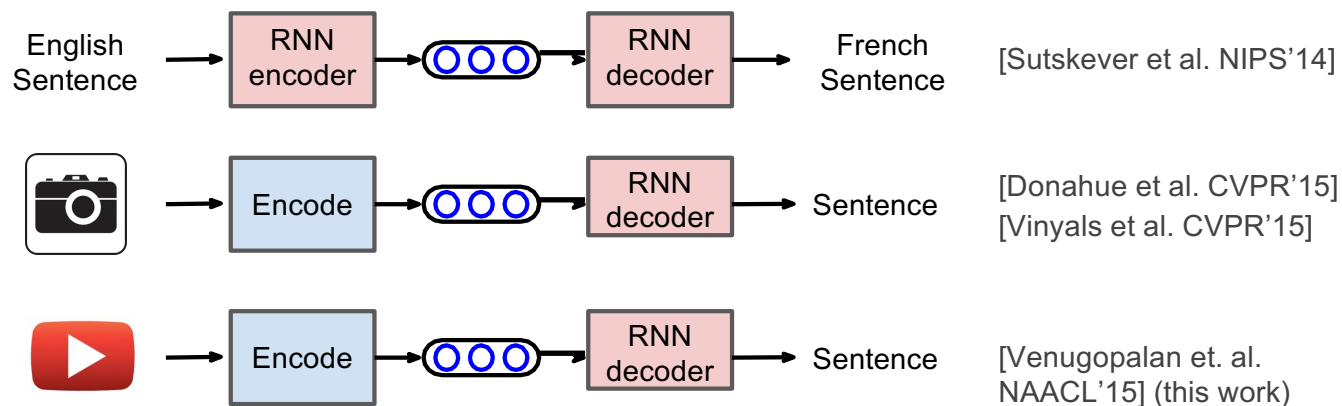
Video Captioning

Generate descriptions for events depicted in video clips



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

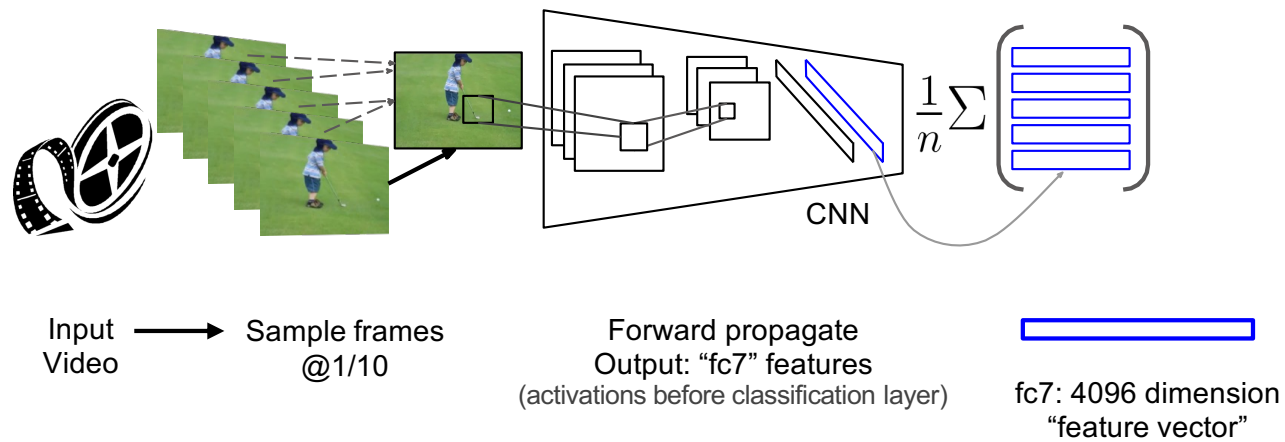
Video Captioning



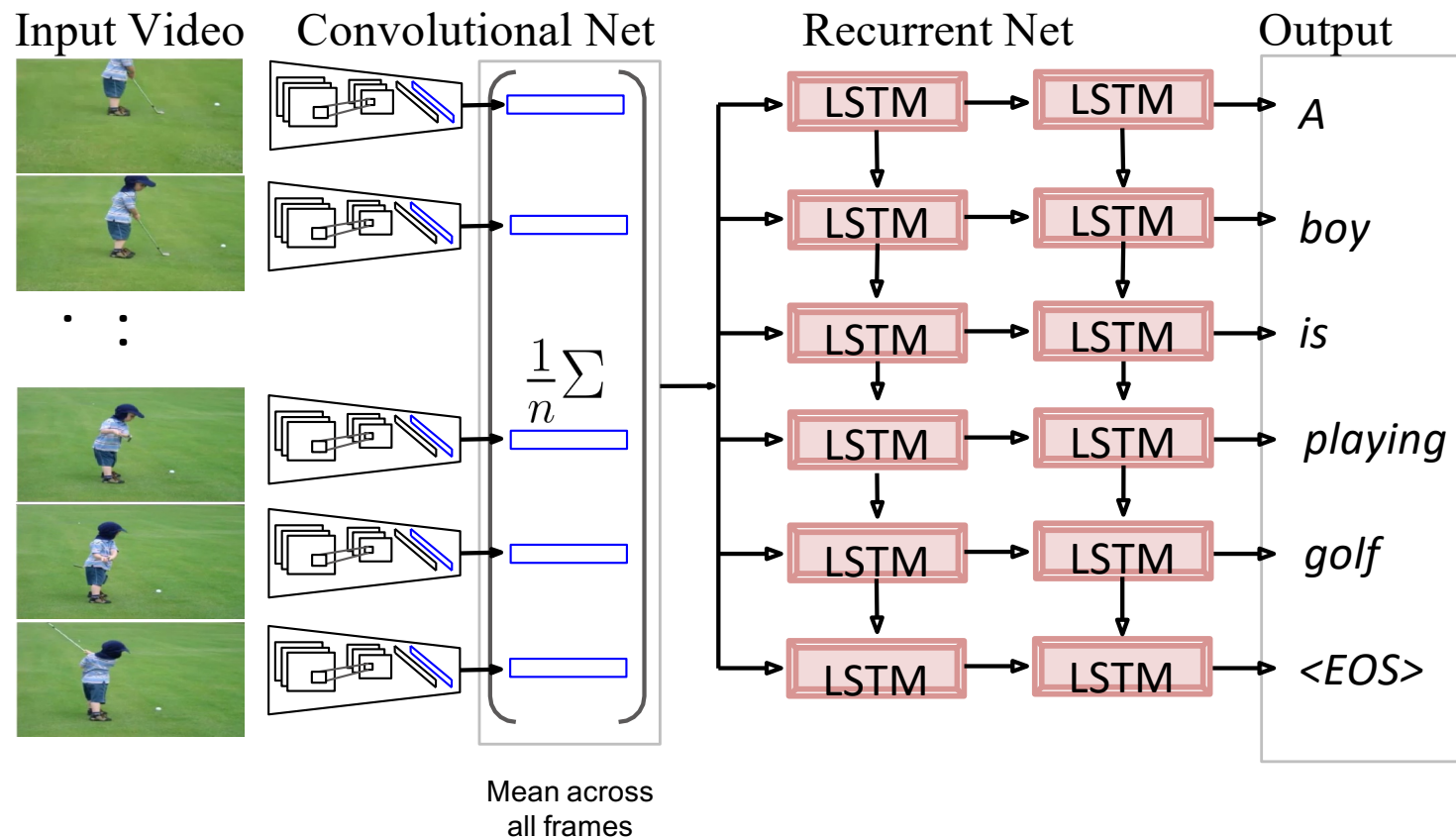
Key Insight:

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

Video Captioning



Video Captioning



Video Captioning

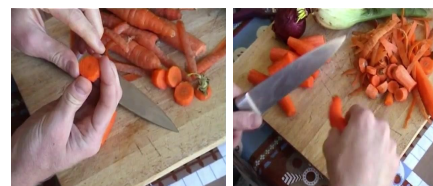


FGM: A person is dancing with the person on the stage.

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.

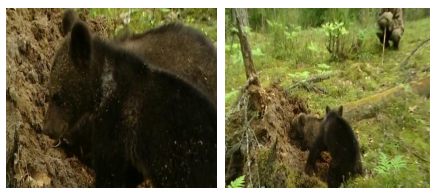


FGM: A person is cutting a potato in the kitchen.

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.



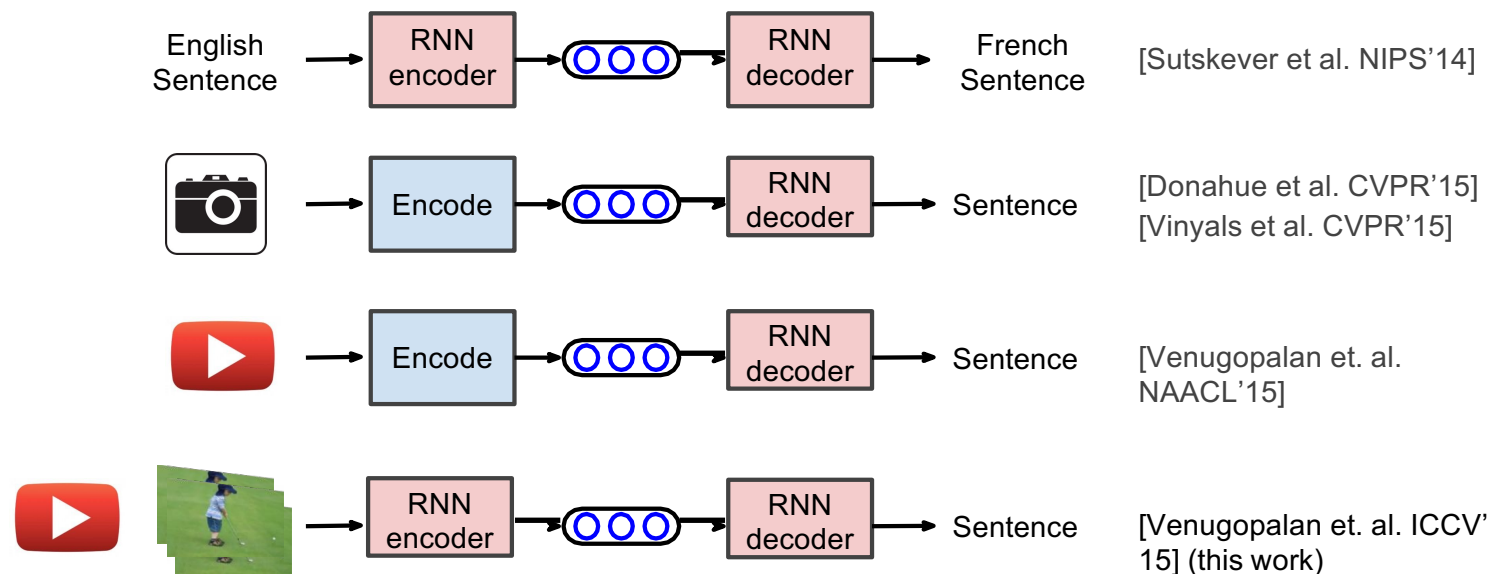
FGM: A person is riding a horse on the stage.

YT: A group of playing are playing in the ball.

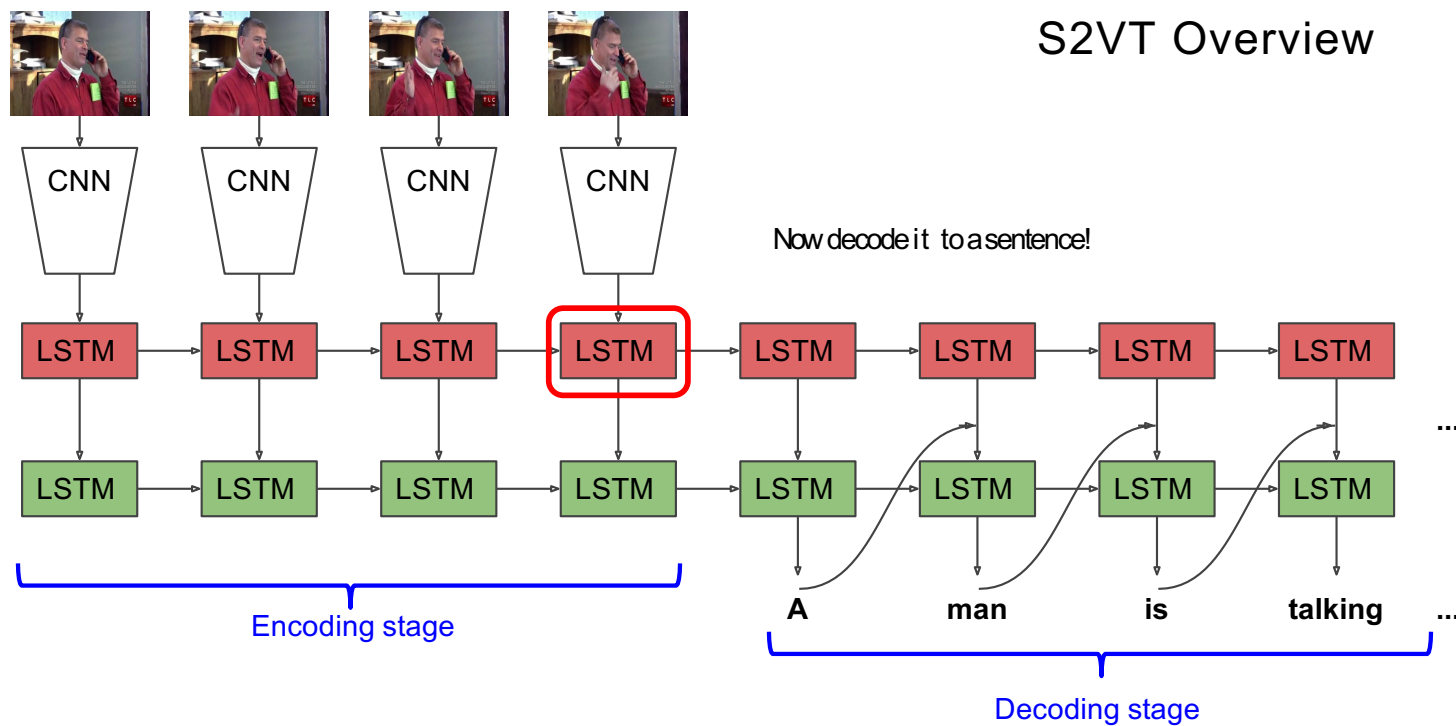
I+V: **A basketball player is playing.**

GT: Dwayne wade does a fancy layup in an allstar game.

Video Captioning



Video Captioning

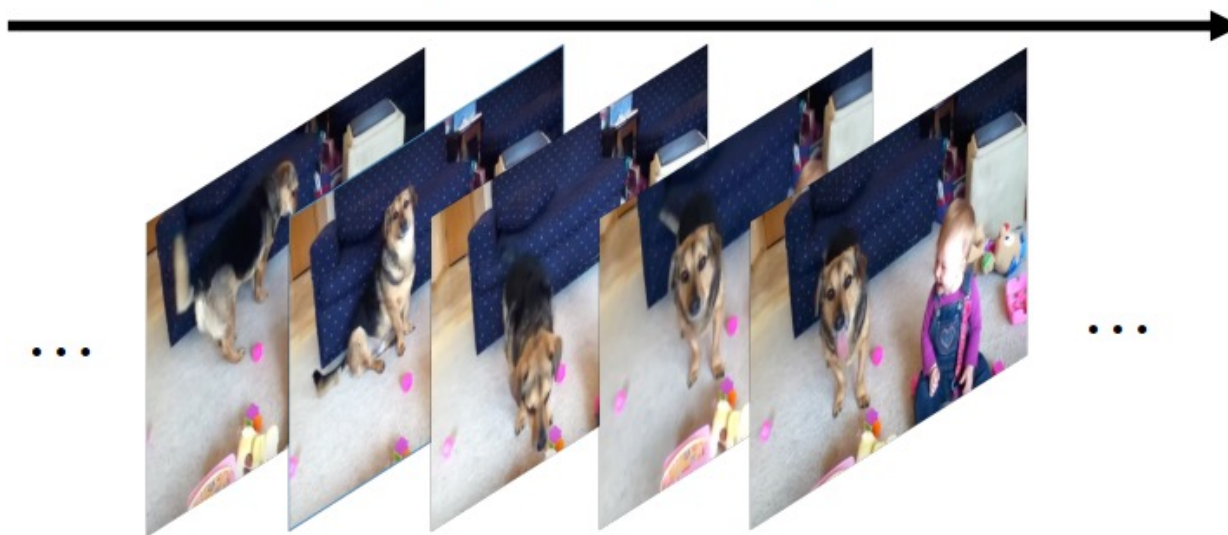


Plan for this lecture

- 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

A video is a sequence of images
4D tensor: $T \times 3 \times H \times W$
(or $3 \times T \times H \times W$)



[This image](#) is [CC0 public domain](#)

Video Classification: Example



Input video:
 $T \times 3 \times H \times W$



Swimming
Running
Jumping
Eating
Standing

Video Classification: Example



Images: Recognize objects



Dog
Cat
Fish
Truck



Videos: Recognize actions



Swimming
Running
Jumping
Eating
Standing

Slide credit: Justin Johnson

Problem: Videos are big!



Input video:
 $T \times 3 \times H \times W$

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)

Training on Clips

Raw video: Long, high FPS



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



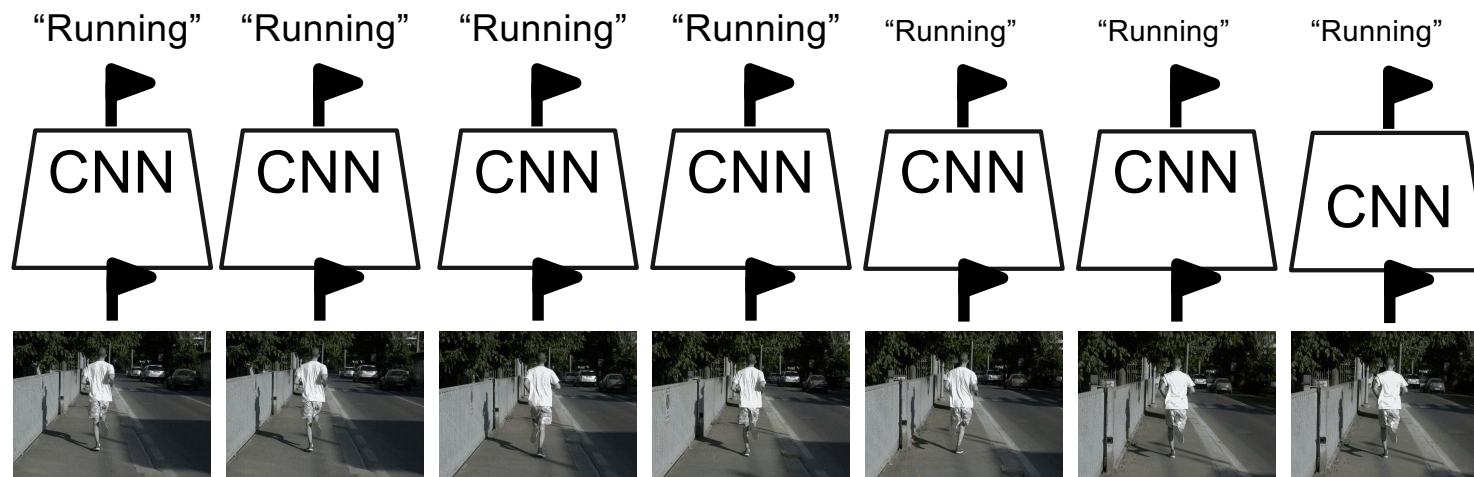
Testing: Run model on different clips, average predictions



Slide credit: Justin Johnson

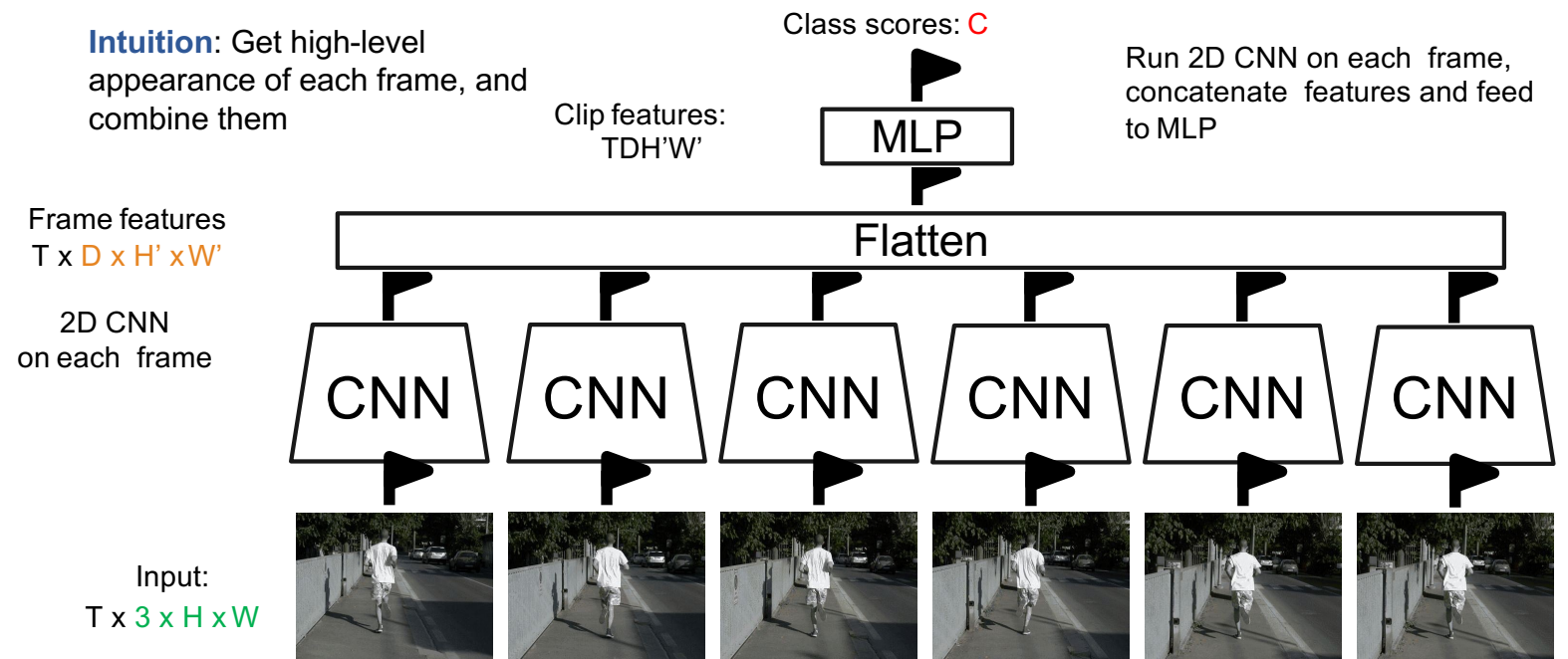
Video Classification: Single-Frame CNN

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



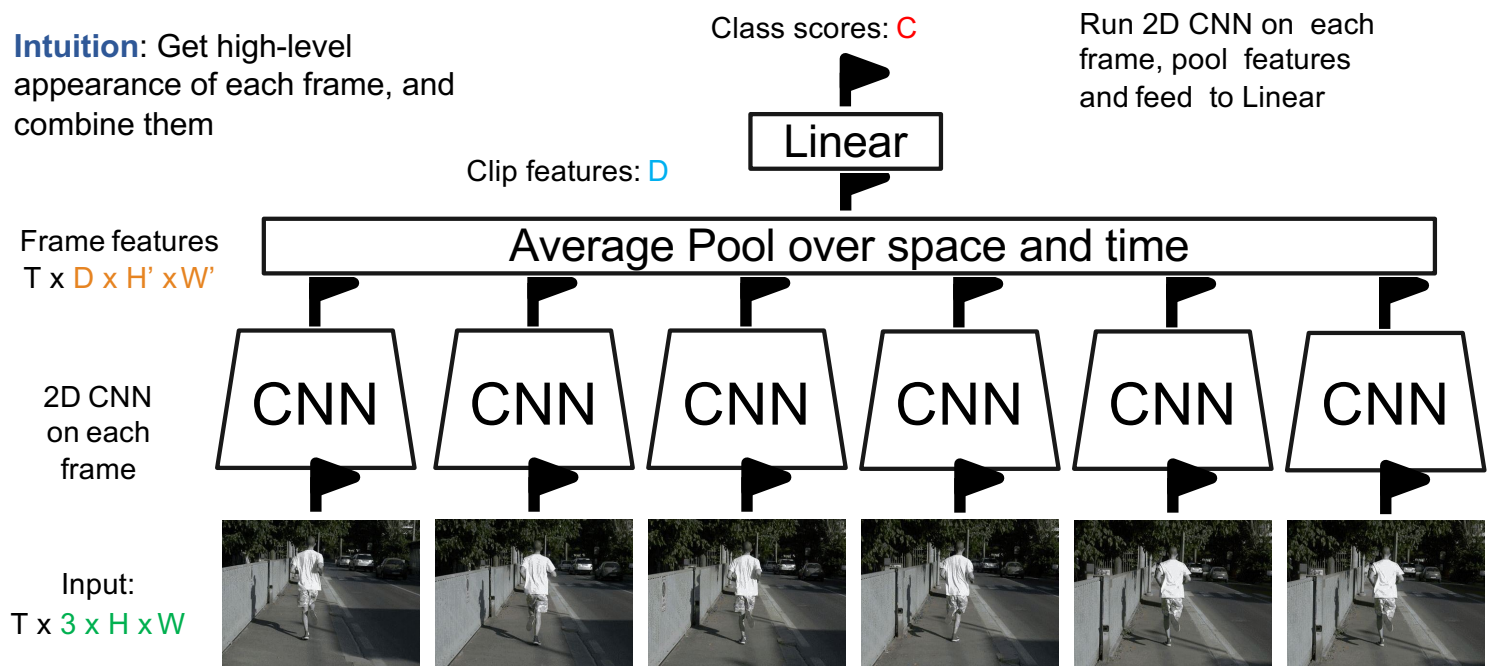
Slide credit: Justin Johnson

Video Classification: Late Fusion (with FC layers)



Video Classification: Late Fusion (with pooling)

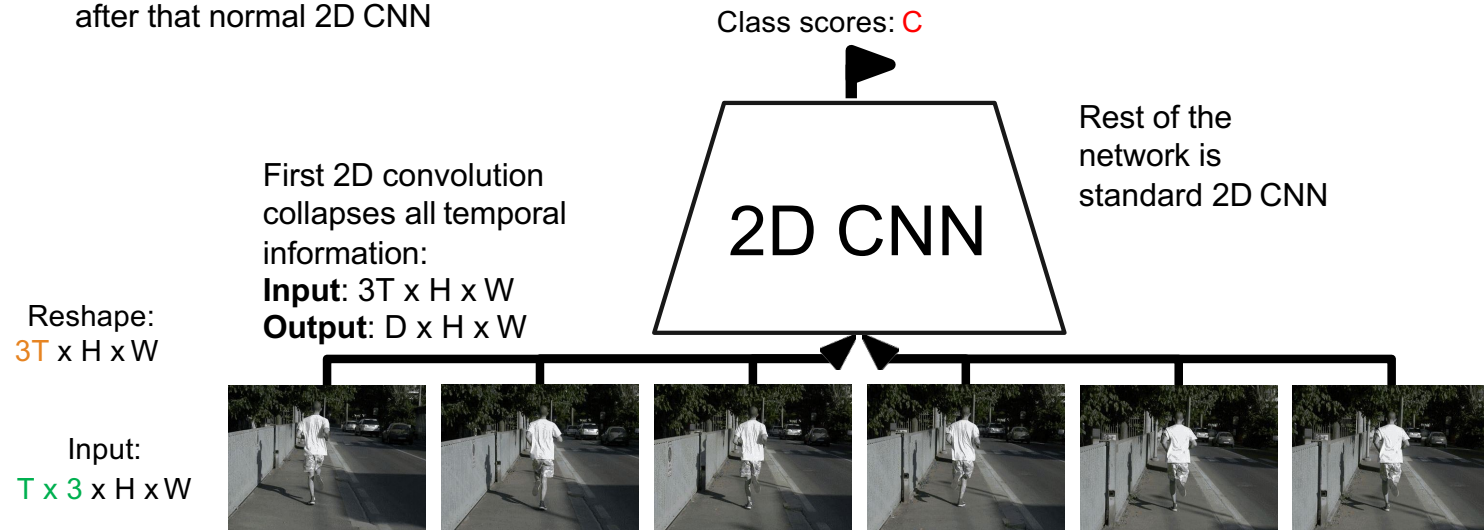
Intuition: Get high-level appearance of each frame, and combine them



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

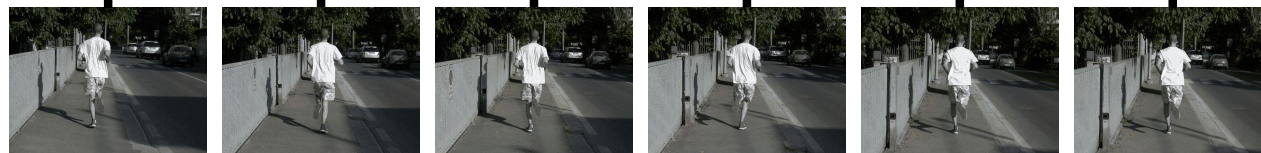
Slide credit: Justin Johnson

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: $D \times T \times H \times W$
Use 3D conv and 3D pooling operations

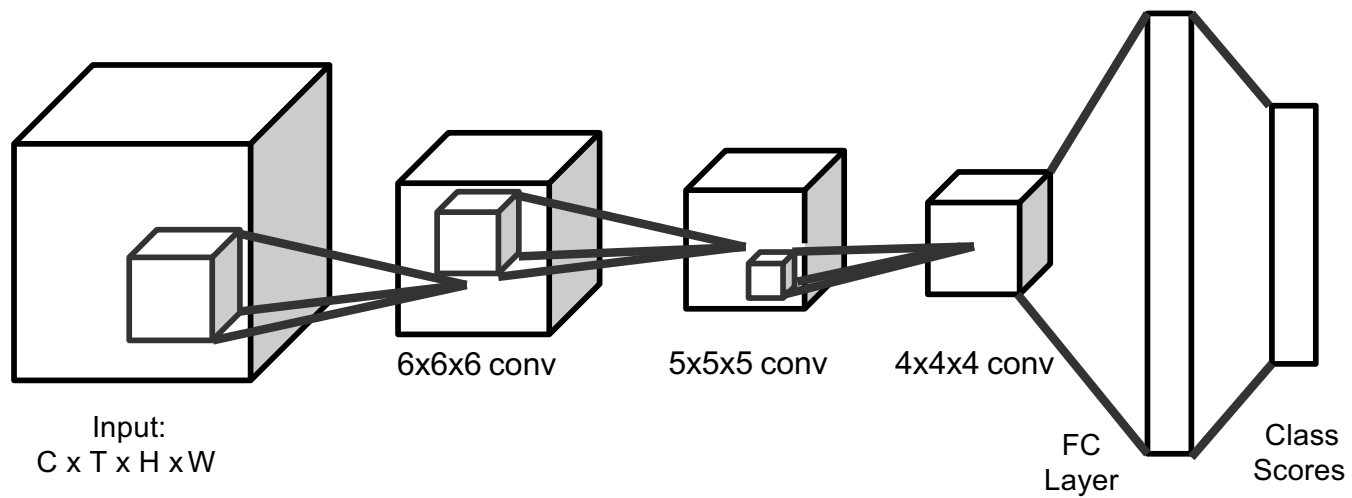
Input:
 $3 \times T \times H \times W$



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



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

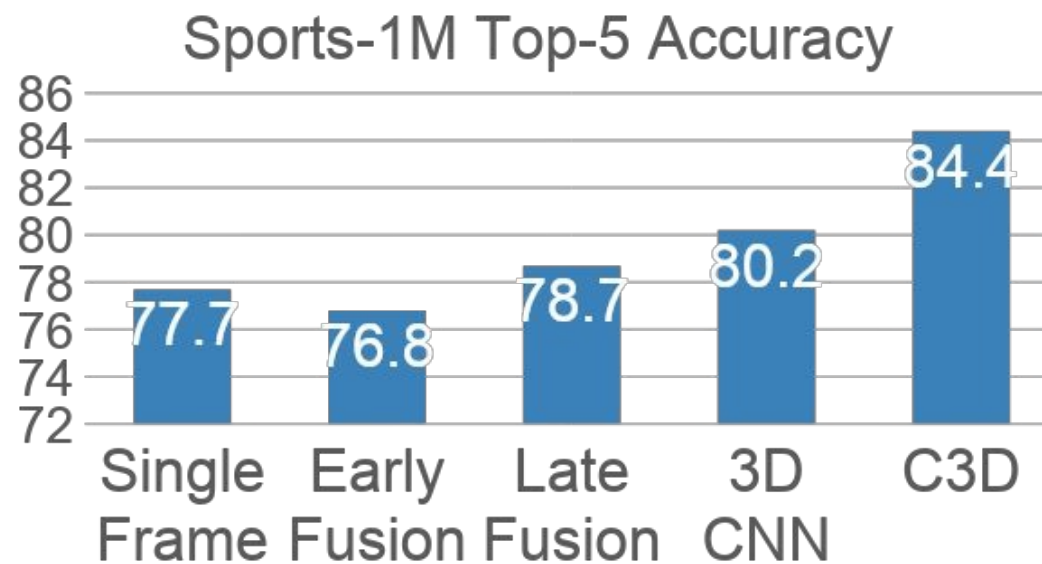
Example Video Dataset: Sports-1M



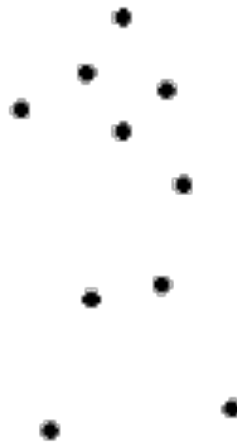
1 million YouTube videos
annotated with labels for 487
different types of sports

Ground Truth
Correct prediction
Incorrect prediction

Early Fusion vs Late Fusion vs 3D CNN



Motion: Why is it useful?



Motion: Why is it useful?

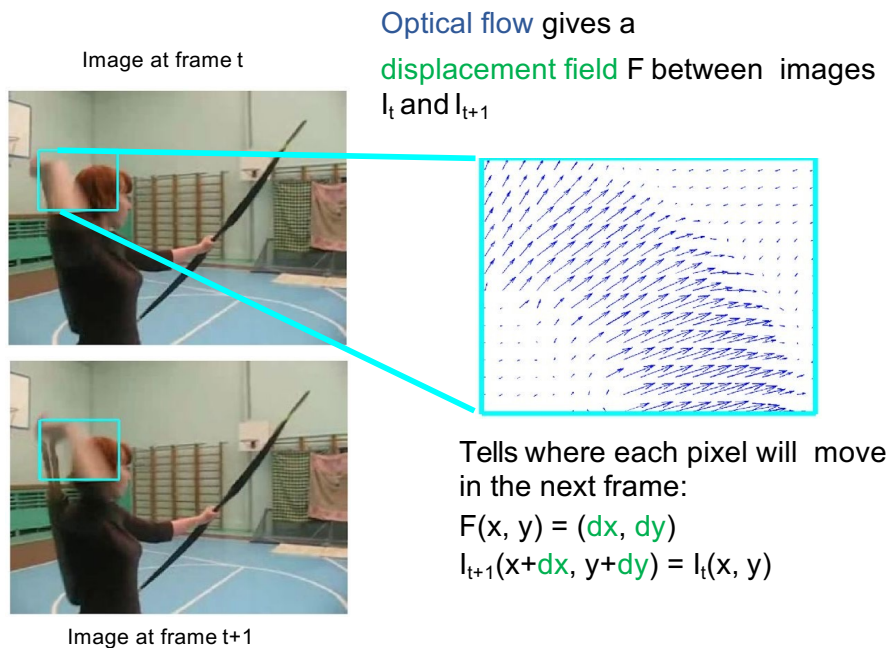
- Even “impoverished” motion data can evoke a strong percept



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

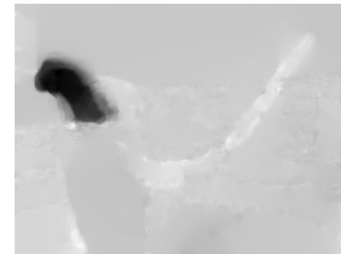
Derek Hoiem

Measuring Motion: Optical Flow



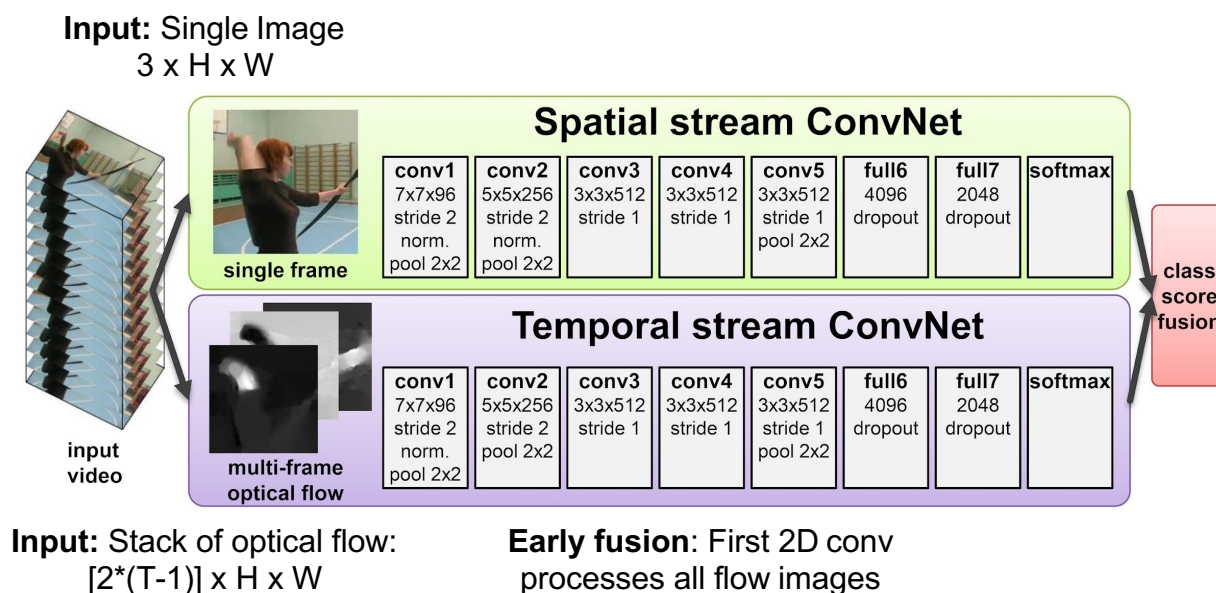
Optical Flow highlights
local motion

Horizontal flow dx

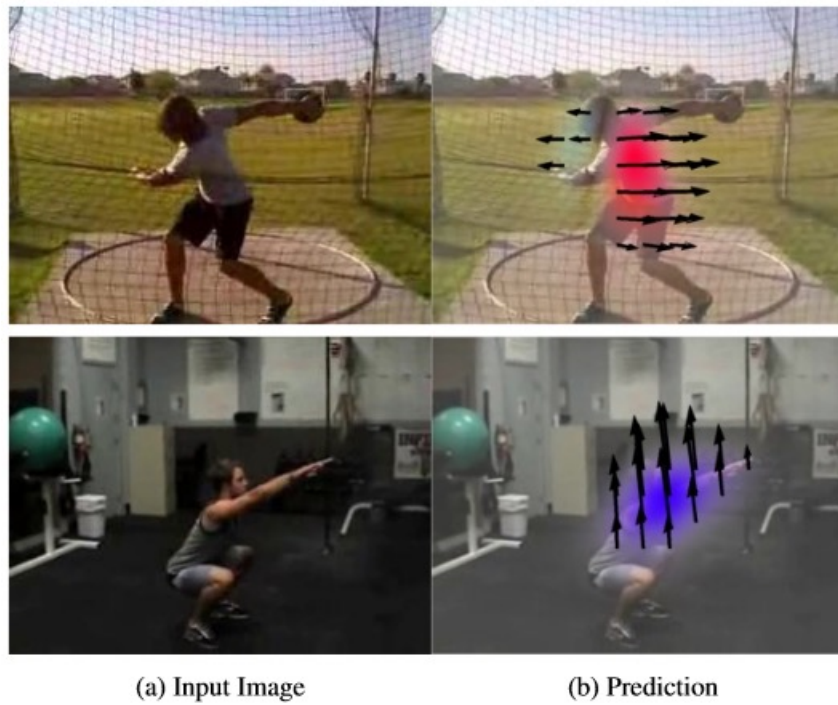


Vertical Flow dy

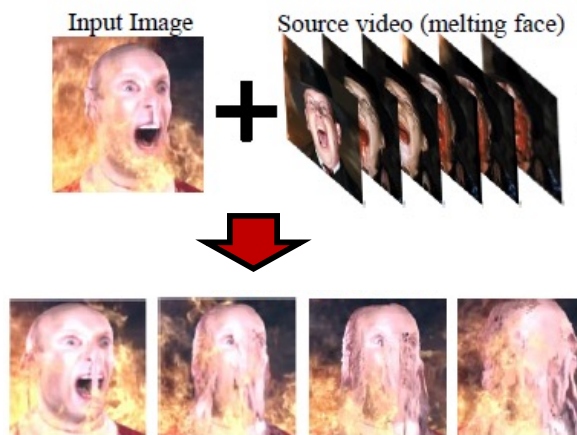
Separating Motion and Appearance: Two-Stream Networks



Modeling Motion: Optical Flow



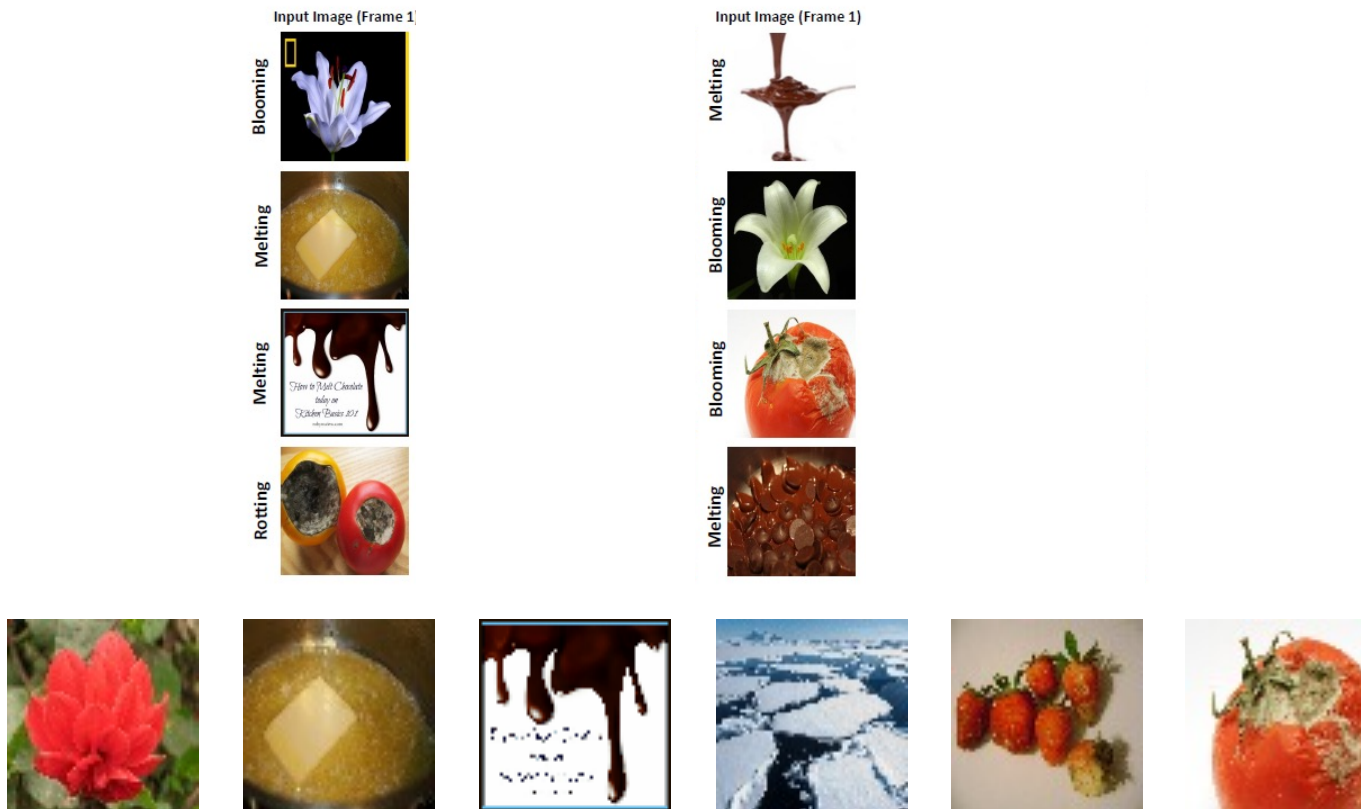
Transferring Motion



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

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

Transferring Motion



Transferring Motion



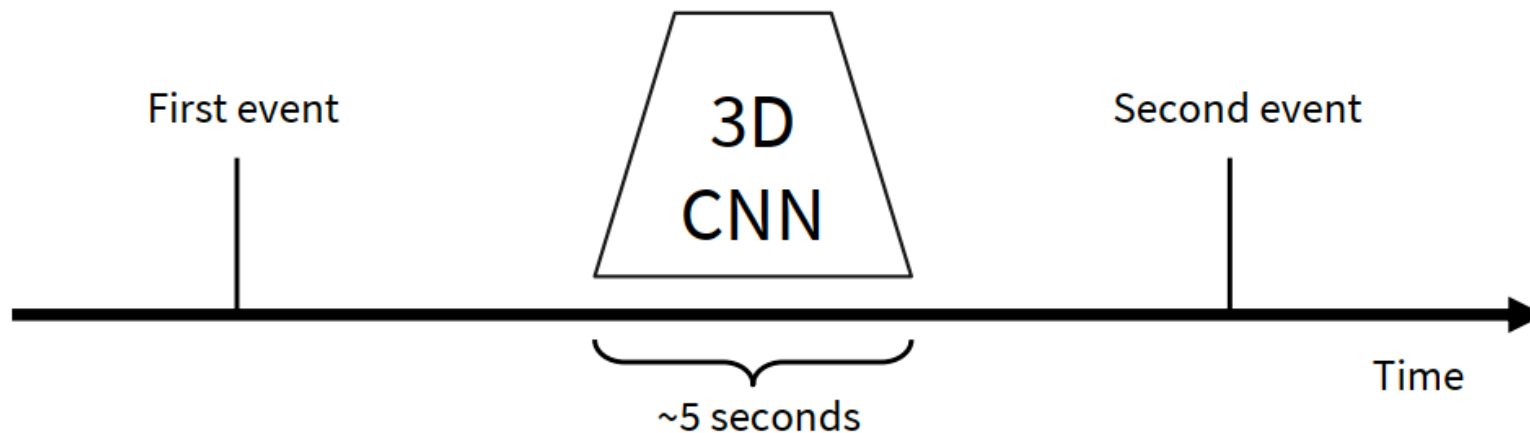
Baking

Blooming

Modeling Long-term Temporal Structure

So far all our temporal CNNs only model local motion between frames in very short clips of ~2-5 seconds. What about long-term structure?

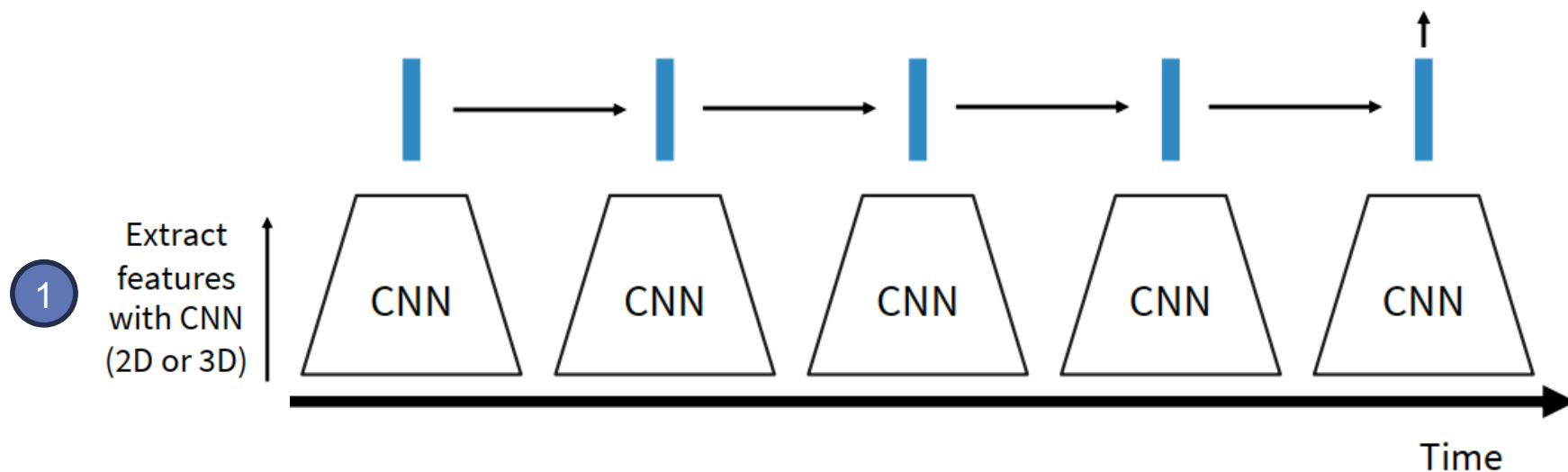
We know how to handle sequences! How about recurrent networks?



Slide credit: Justin Johnson

Modeling Long-term Temporal Structure

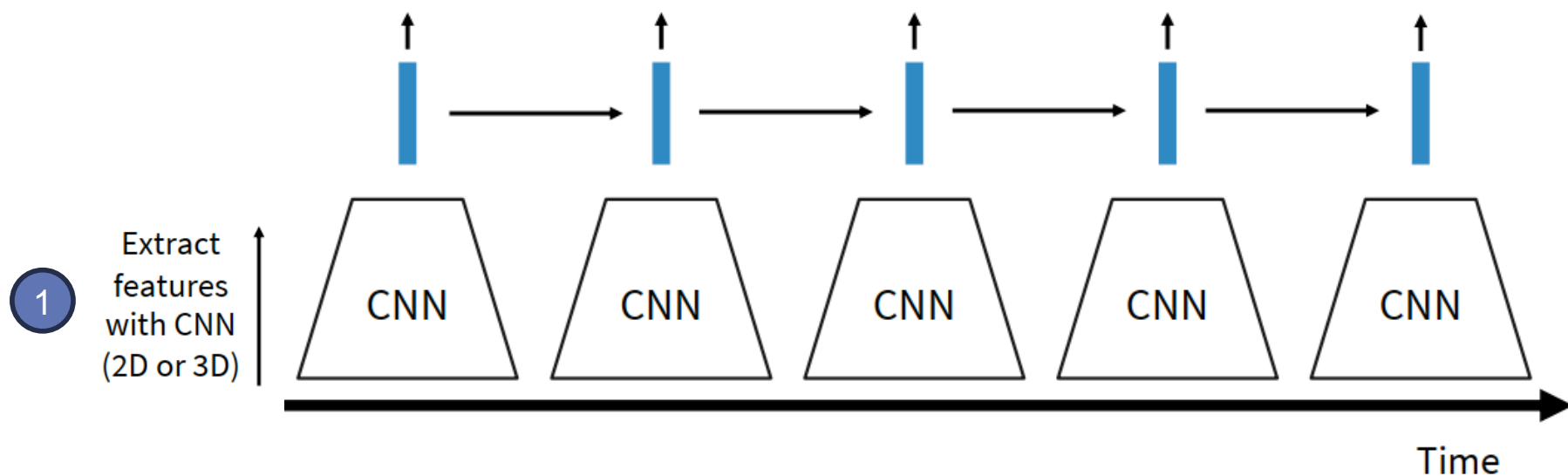
- 2 Process local features using recurrent network (e.g. LSTM)
- 3 Many to one: One output at end of video



Slide credit: Justin Johnson

Modeling Long-term Temporal Structure

- 2 Process local features using recurrent network (e.g. LSTM)
- 3 Many to many: one output per video frame



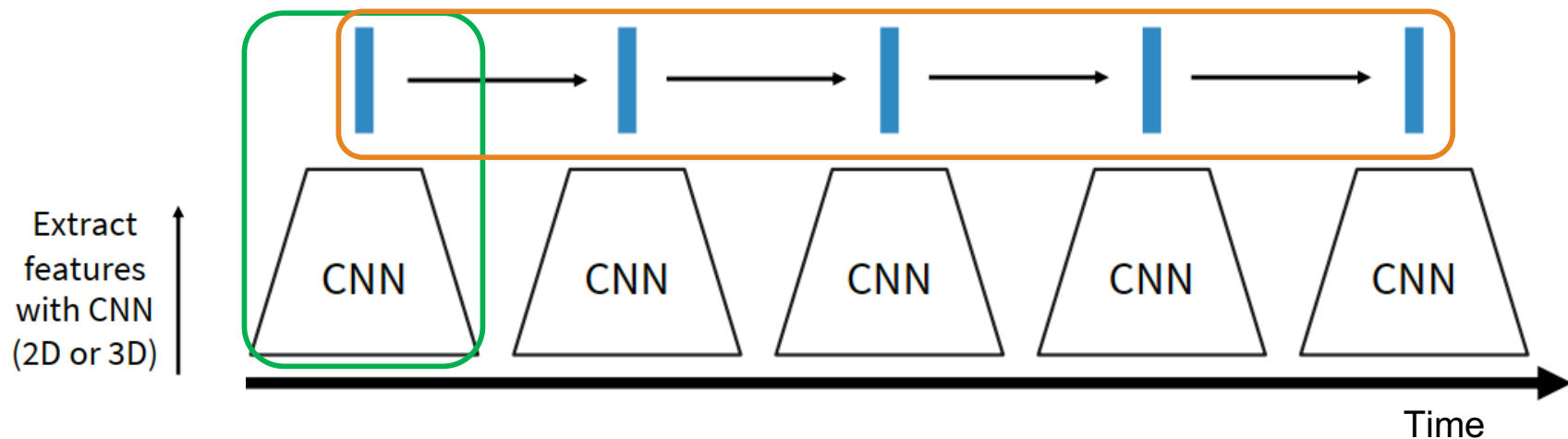
Slide credit: Justin Johnson

Modeling Long-term Temporal Structure

Inside CNN: Each value is a function of a fixed temporal window (local temporal structure)

Inside RNN: Each vector is a function of all previous vectors (global temporal structure)

Can we merge both approaches?



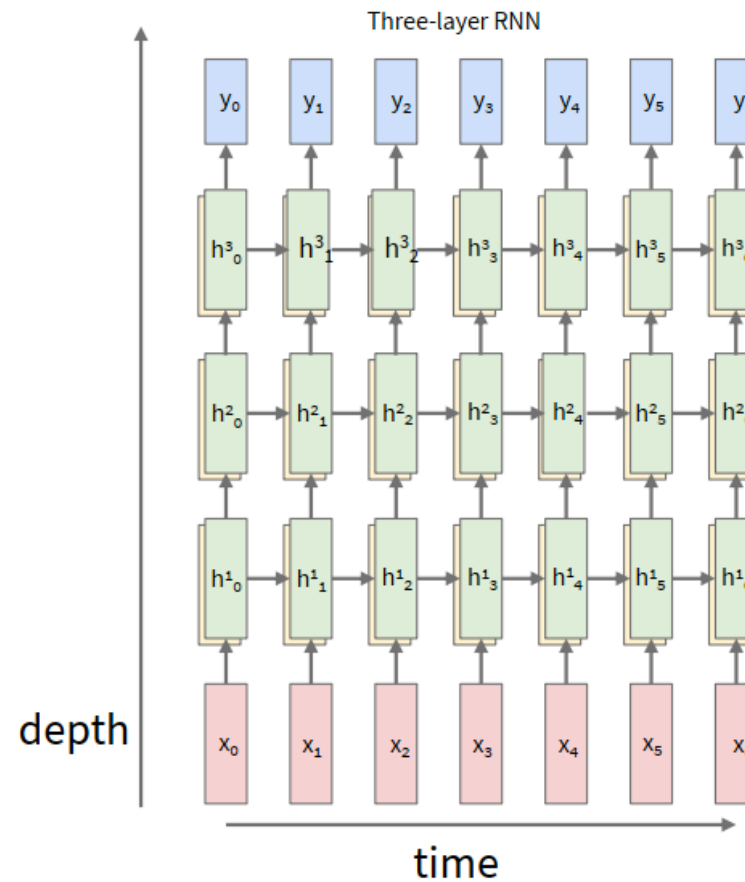
Baccouche et al, "Sequential Deep Learning for Human Action Recognition", 2011

Donahue et al, "Long-term recurrent convolutional networks for visual recognition and description", CVPR 2015

Slide credit: Justin Johnson

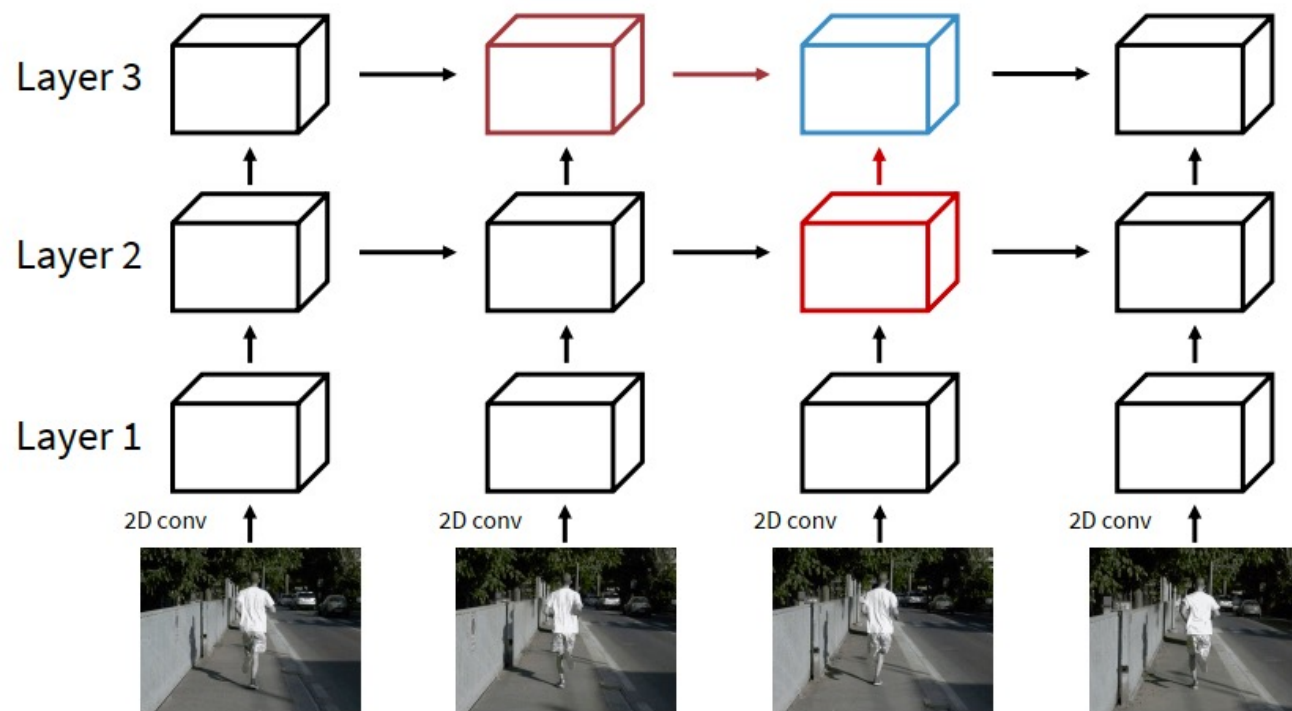
Intuition: Multi-layer RNN

We can use a similar structure to process videos!



Slide credit: Justin Johnson

Recurrent Convolutional Network



Entire network
uses 2D feature
maps: $C \times H \times W$

Each depends on
two inputs:
1. Same layer,
previous timestep
2. Prev layer,
same timestep

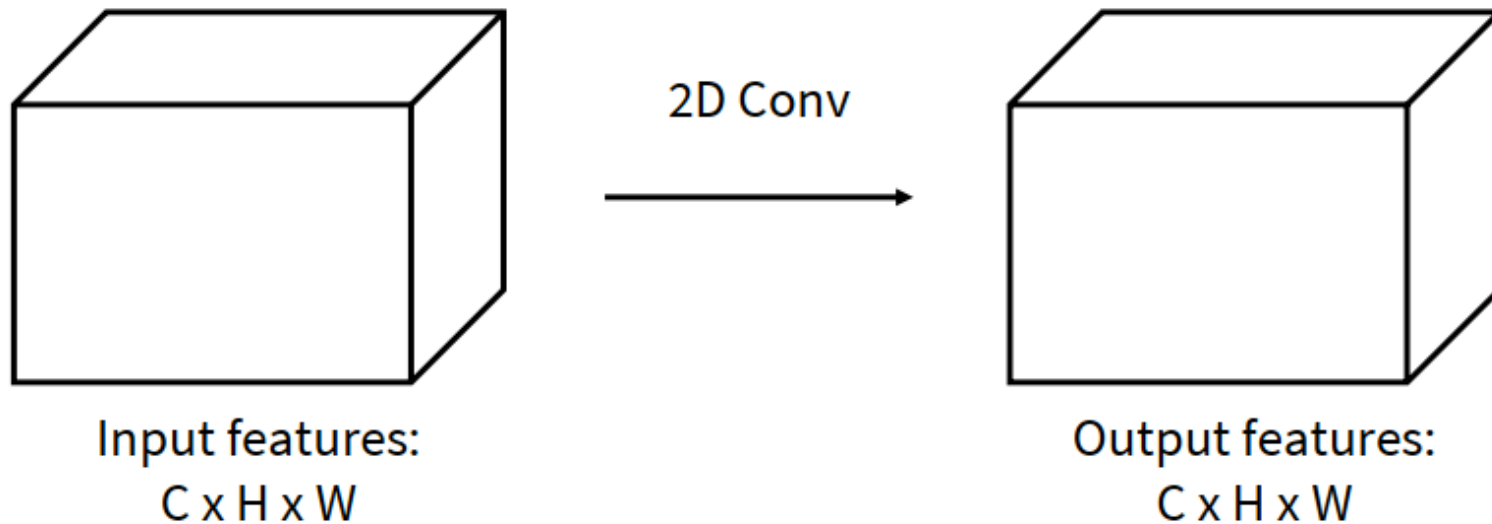
Use different weights
at each layer, share
weights across time

Ballas et al, "Delving Deeper into
Convolutional Networks for Learning
Video Representations", ICLR 2016

Slide credit: Justin Johnson

Recurrent Convolutional Network

Normal 2D CNN:



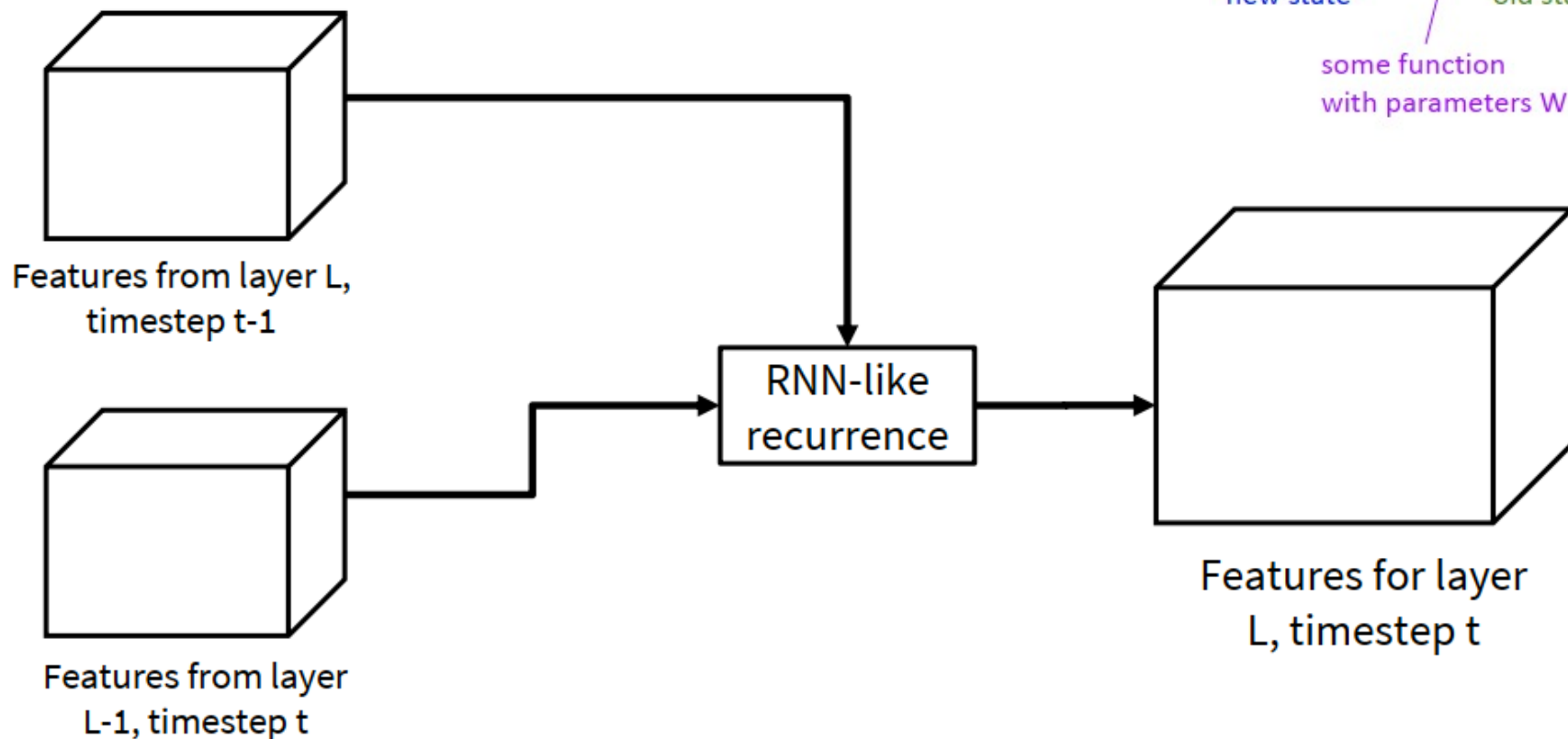
Slide credit: Justin Johnson

Recurrent Convolutional Network

Recall: Recurrent Network

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

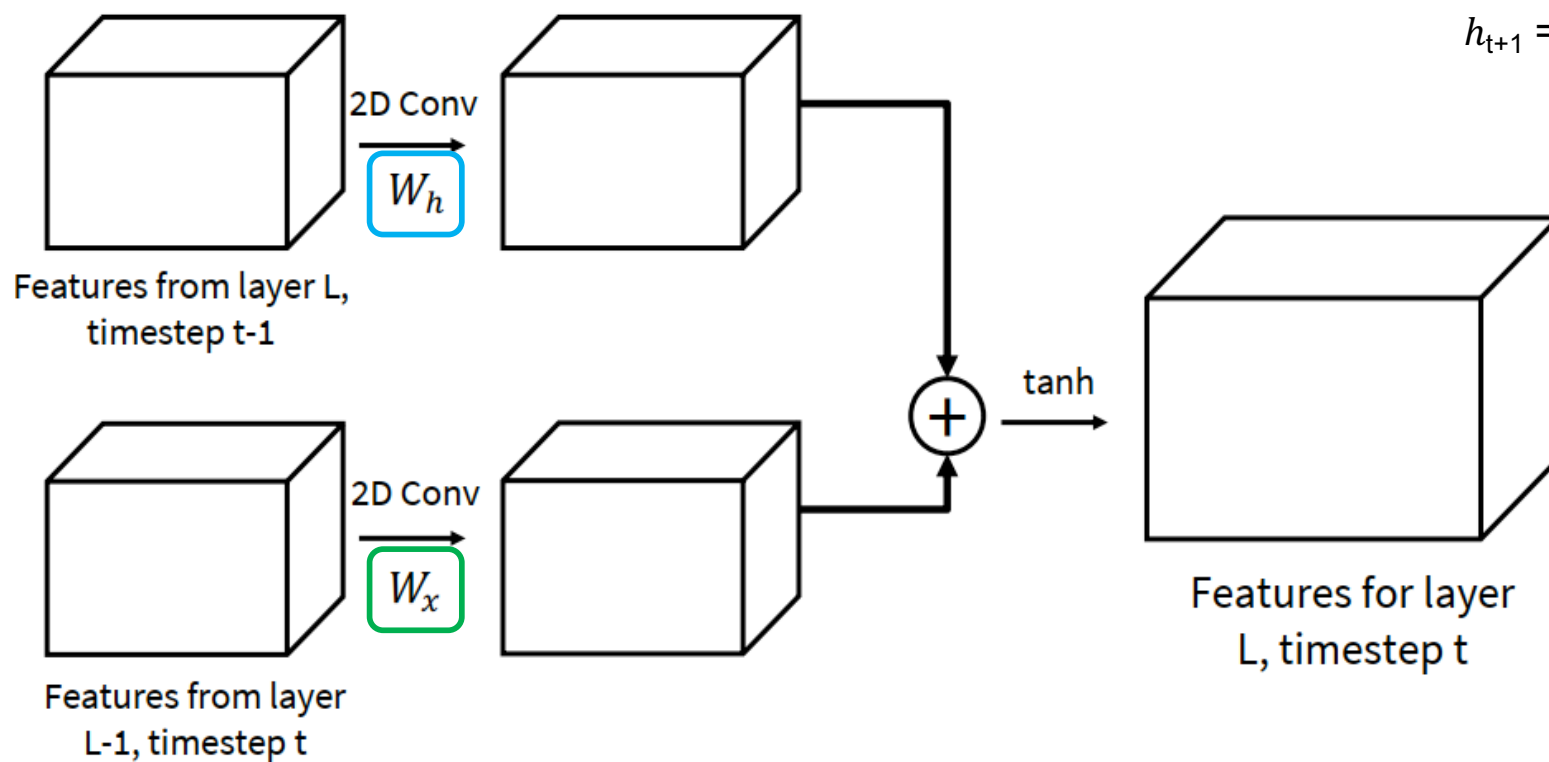
new state / old state
some function with parameters W



Recurrent Convolutional Network

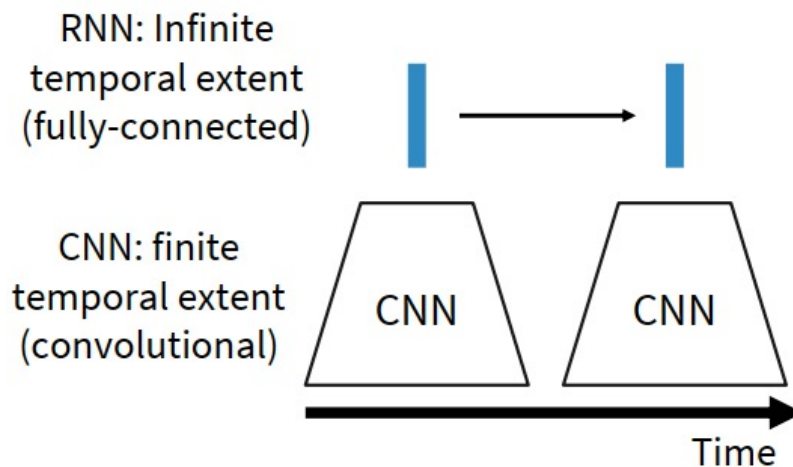
Recall: Vanilla RNN

$$h_{t+1} = \tanh(W_h h_t + W_x x)$$



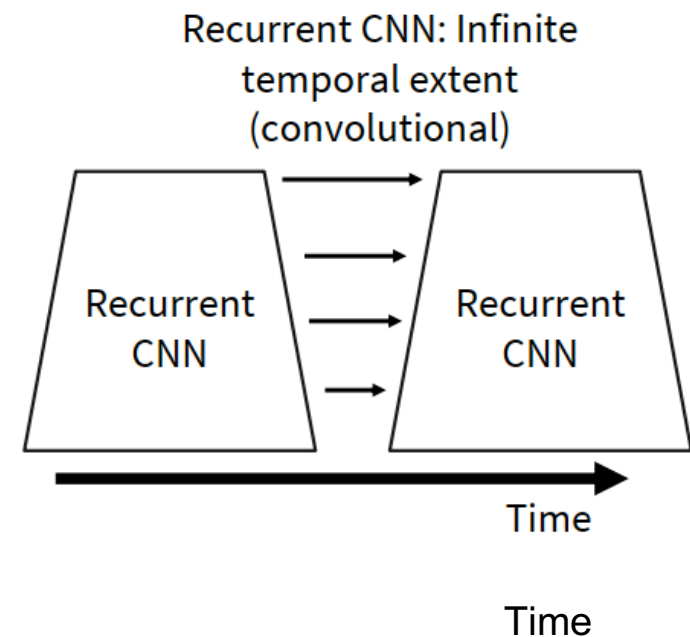
Modeling Long-term Temporal Structure

Problem: RNNs are slow for long sequences (can't be parallelized)



Baccouche et al, "Sequential Deep Learning for Human Action Recognition", 2011

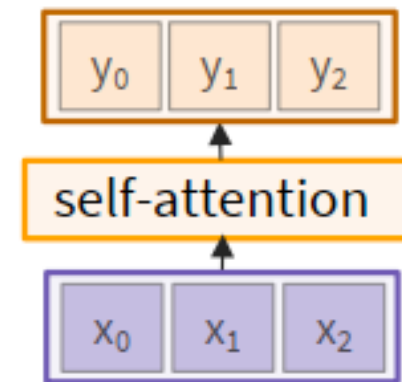
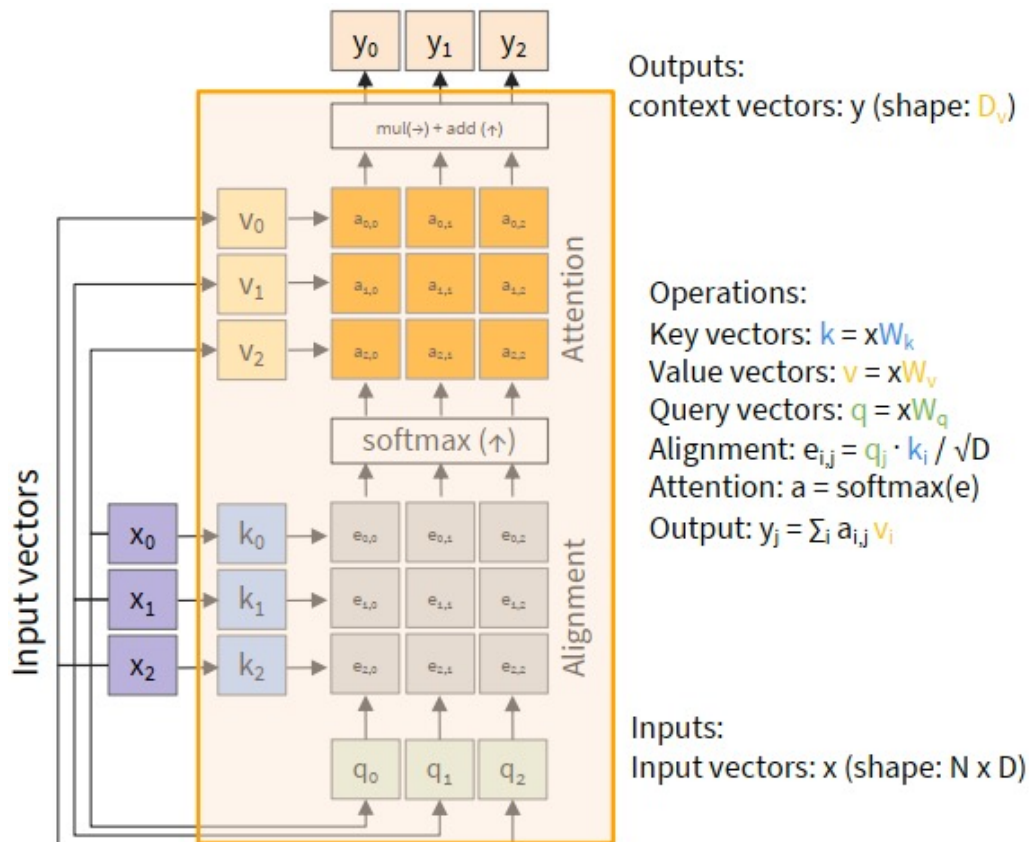
Donahue et al, "Long-term recurrent convolutional networks for visual recognition and description", CVPR 2015



Ballas et al, "Delving Deeper into Convolutional Networks for Learning Video Representations", ICLR 2016

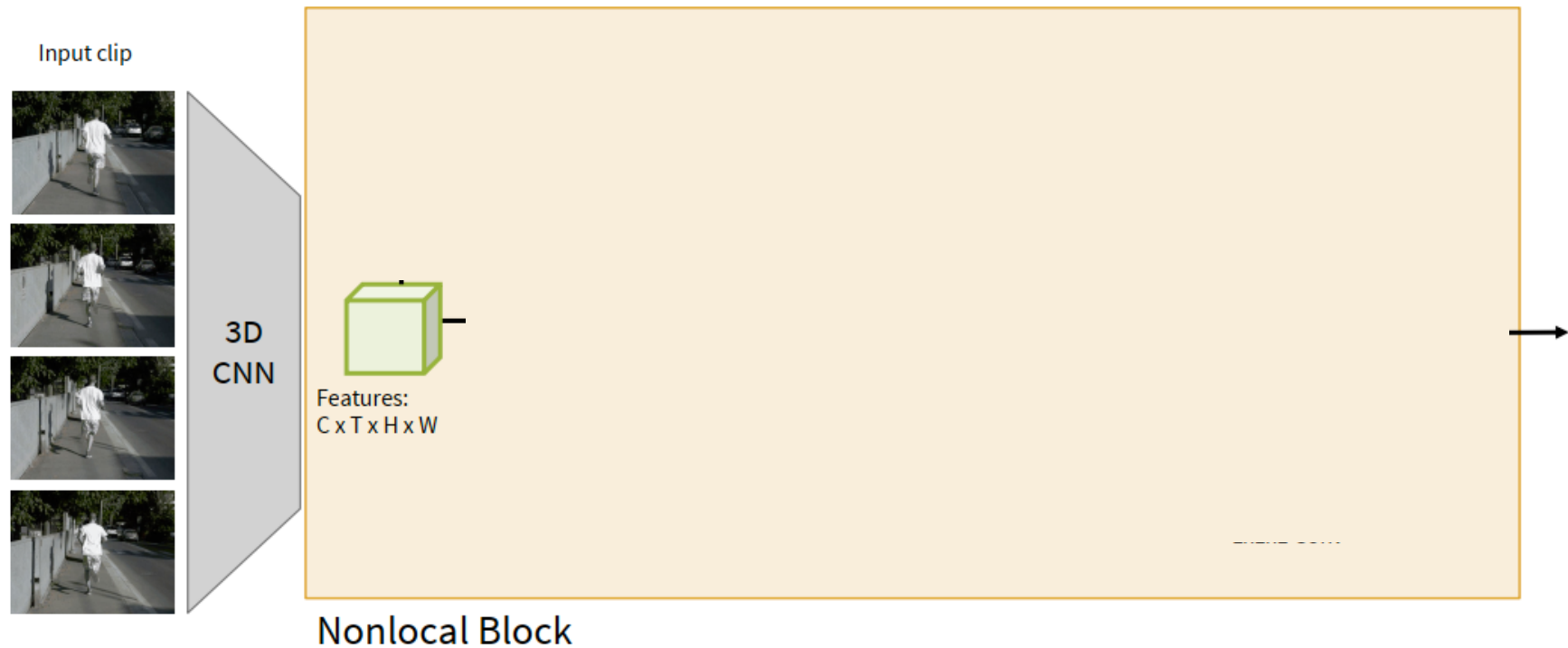
Slide credit: Justin Johnson

Recall: Self-Attention

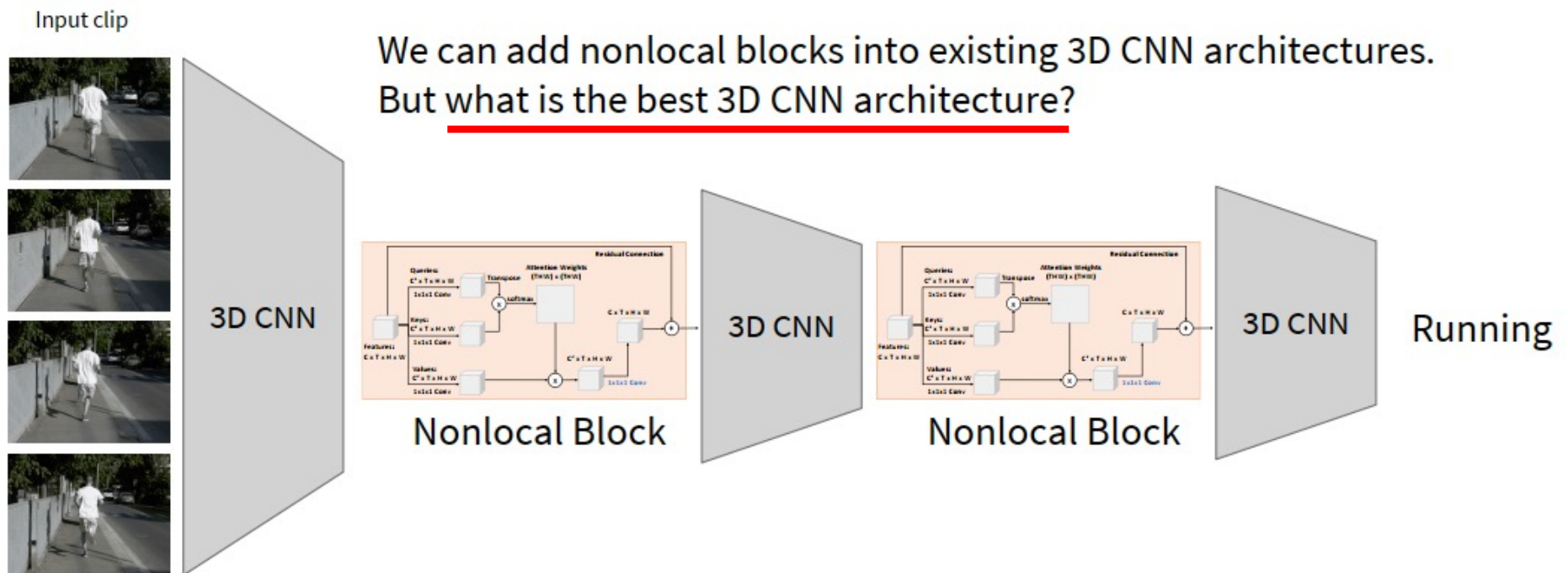


Slide credit: Fei-Fei Li

Spatio-Temporal Self-Attention (Nonlocal Block)



Spatio-Temporal Self-Attention (Nonlocal Block)

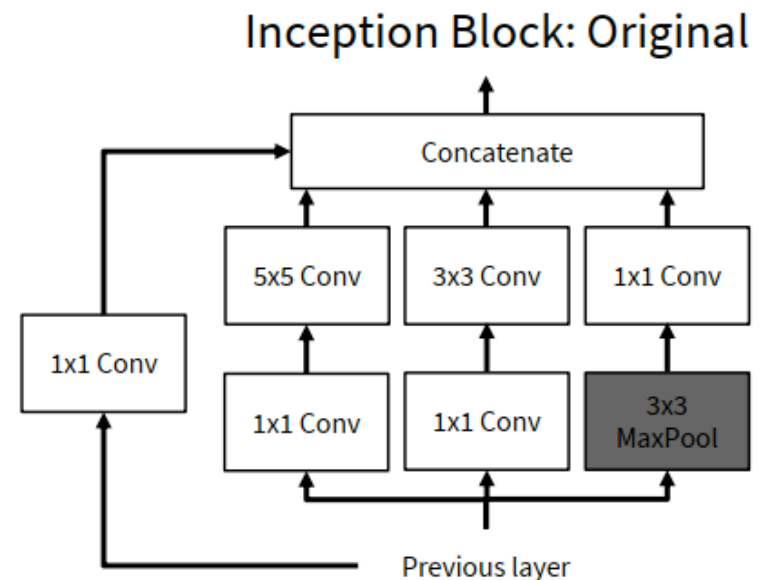


Inflating 2D Networks to 3D (I3D)

There has been a lot of work on architectures for images.
Can we reuse image architectures for video?

Idea: take a 2D CNN architecture.

Replace each 2D $K_h \times K_w$ conv/pool layer with a 3D $K_t \times K_h \times K_w$ version

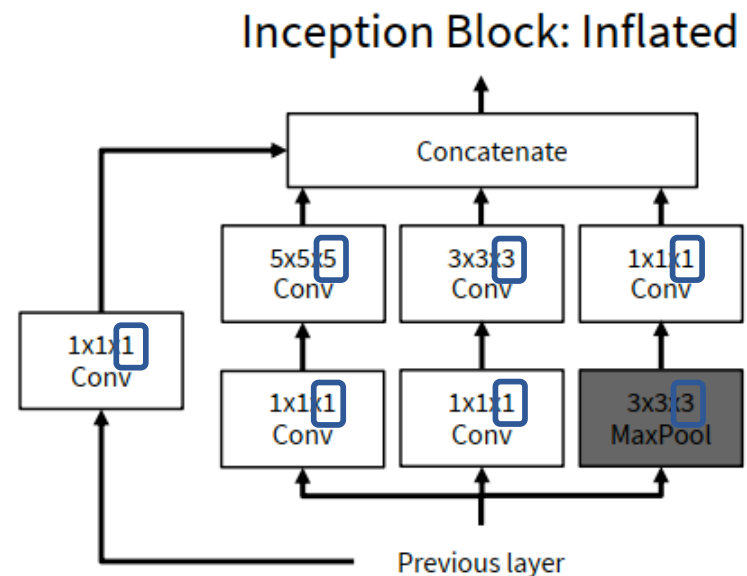


Inflating 2D Networks to 3D (I3D)

There has been a lot of work on architectures for images.
Can we reuse image architectures for video?

Idea: take a 2D CNN architecture.

Replace each 2D $K_h \times K_w$ conv/pool layer with a 3D $K_t \times K_h \times K_w$ version



Inflating 2D Networks to 3D (I3D)

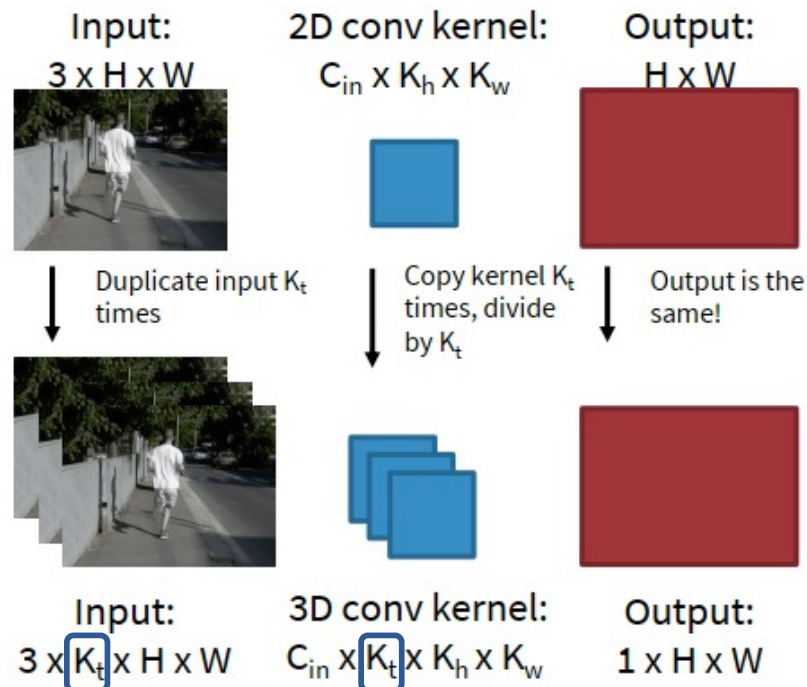
There has been a lot of work on architectures for images.
Can we reuse image architectures for video?

Idea: take a 2D CNN architecture.

Replace each 2D $K_h \times K_w$ conv/pool layer with a 3D $K_t \times K_h \times K_w$ version

Can use weights of 2D conv to initialize 3D conv: copy K_t times in space and divide by K_t

This gives the same result as 2D conv given “constant” video input



Inflating 2D Networks to 3D (I3D)

There has been a lot of work on architectures for images.
Can we reuse image architectures for video?

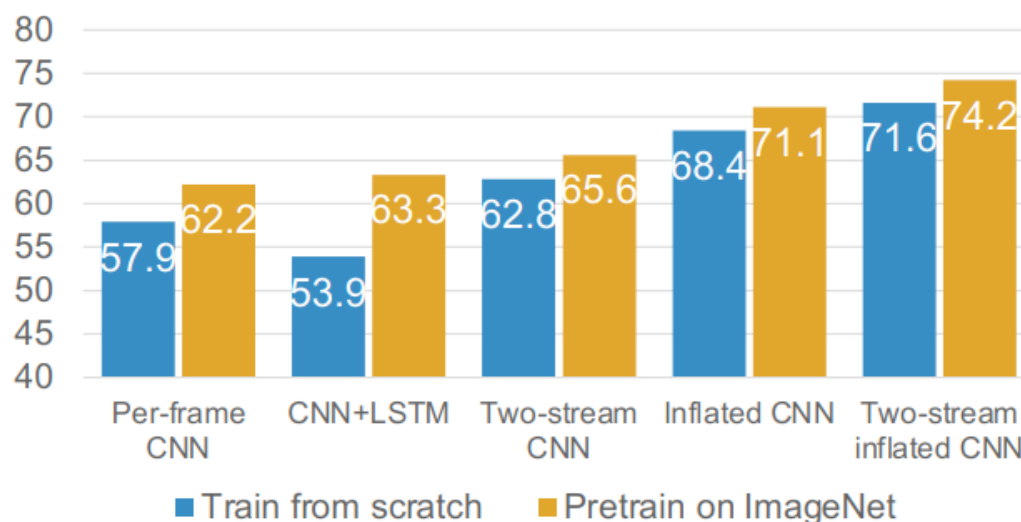
Idea: take a 2D CNN architecture.

Replace each 2D $K_h \times K_w$ conv/pool layer with a 3D $K_t \times K_h \times K_w$ version

Can use weights of 2D conv to initialize 3D conv: copy K_t times in space and divide by K_t

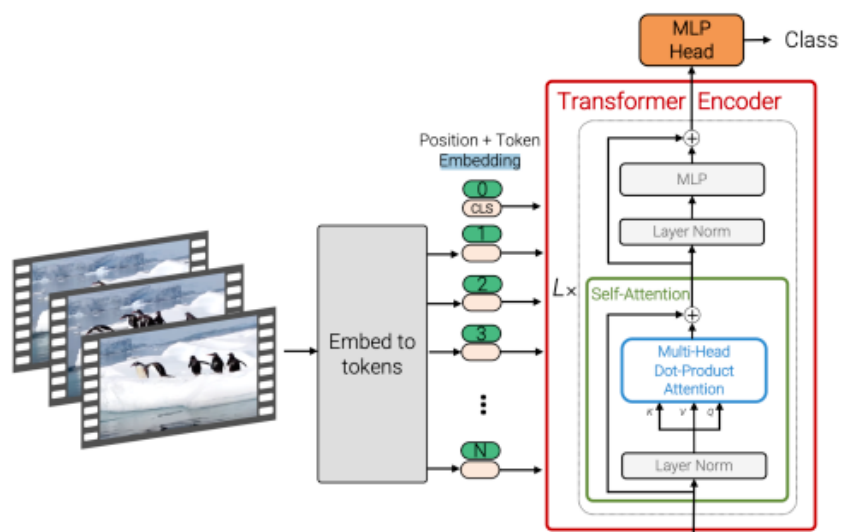
This gives the same result as 2D conv given “constant” video input

Top-1 Accuracy on Kinetics-400



Vision Transformers for Video

Factorized attention: Attend over space / time

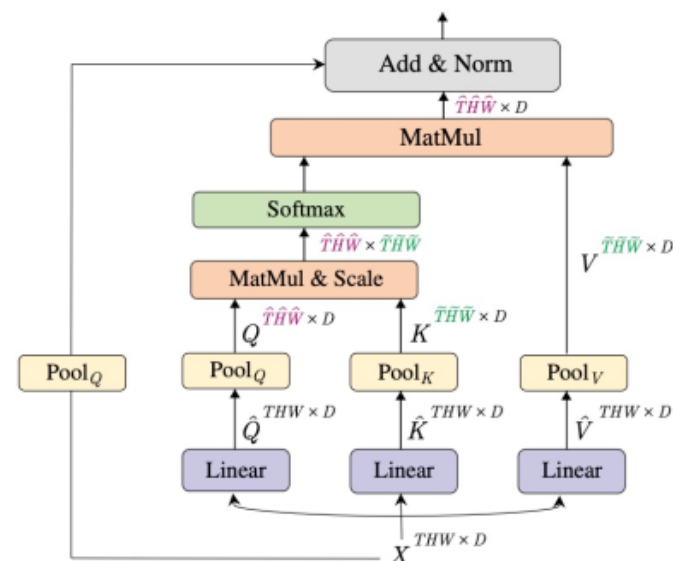


Bertasius et al, "Is Space-Time Attention All You Need for Video Understanding?", ICML 2021

Arnab et al, "ViViT: A Video Vision Transformer", ICCV 2021

Neimark et al, "Video Transformer Network", ICCV 2021

Pooling module: Reduce number of tokens

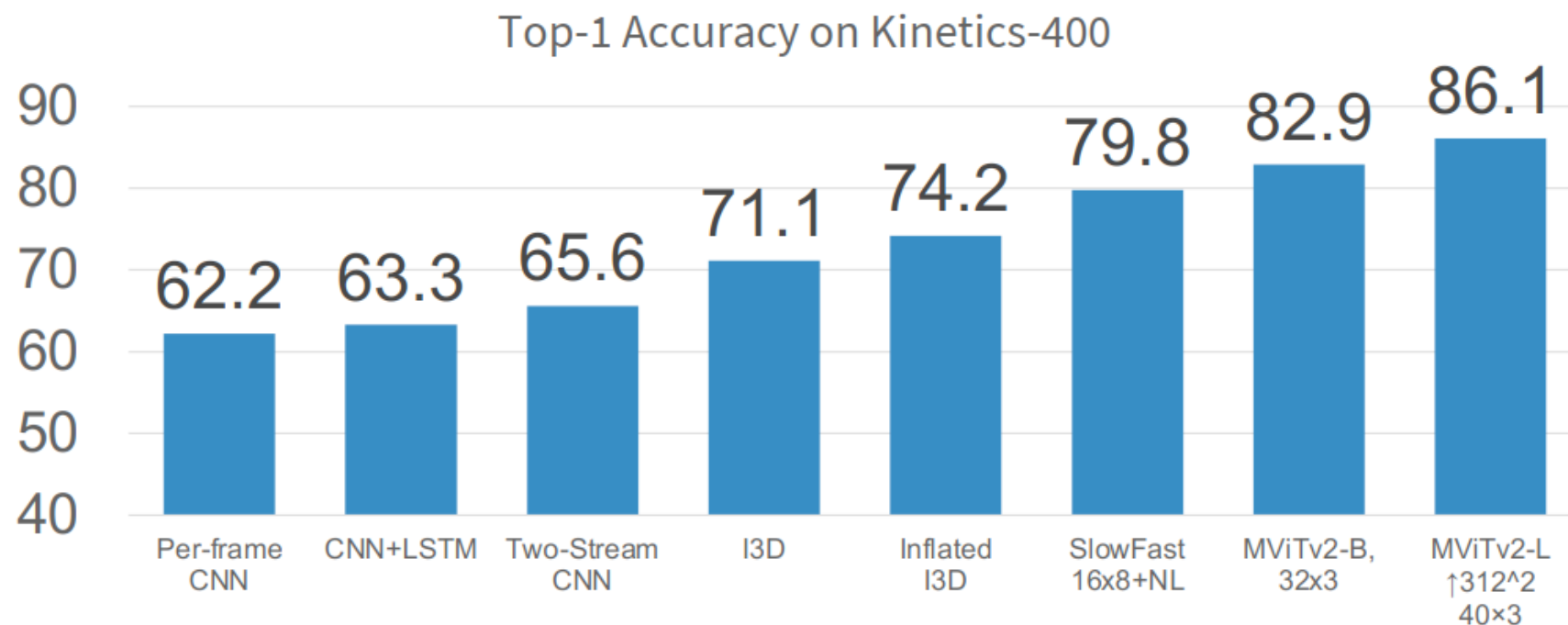


Fan et al, "Multiscale Vision Transformers", ICCV 2021

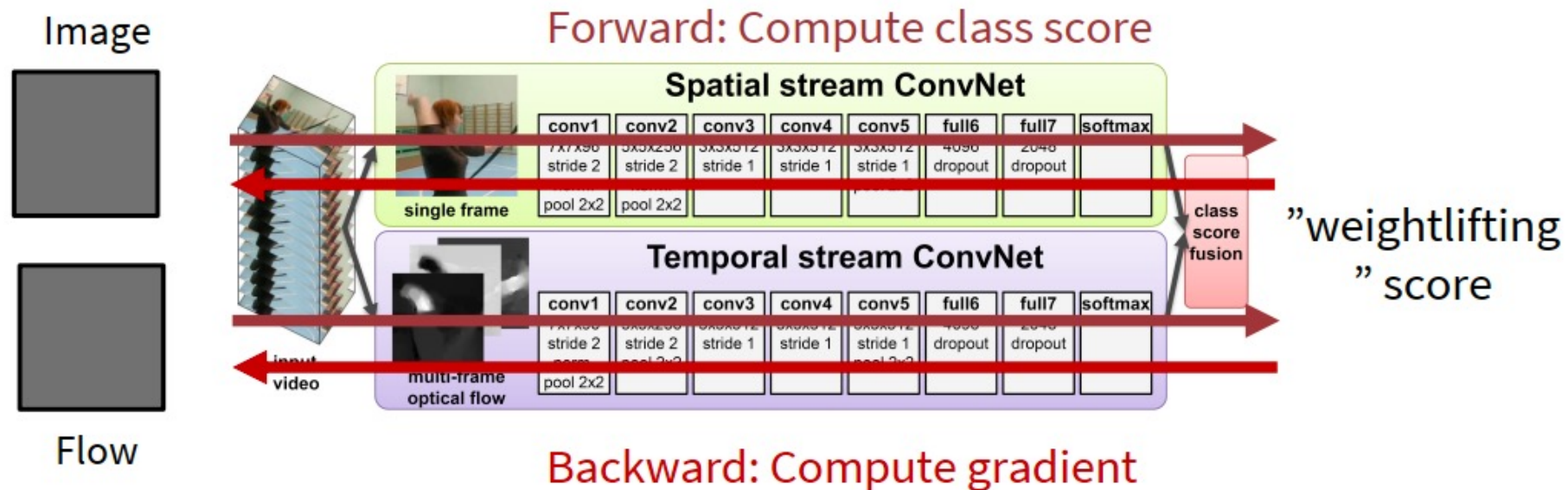
Li et al, "MViTv2: Improved Multiscale Vision Transformers for Classification and Detection", CVPR 2022

Slide credit: Fei-Fei Li

Vision Transformers for Video



Visualizing Video Models



Add a term to encourage spatially smooth flow; tune penalty to pick out "slow" vs "fast" motion

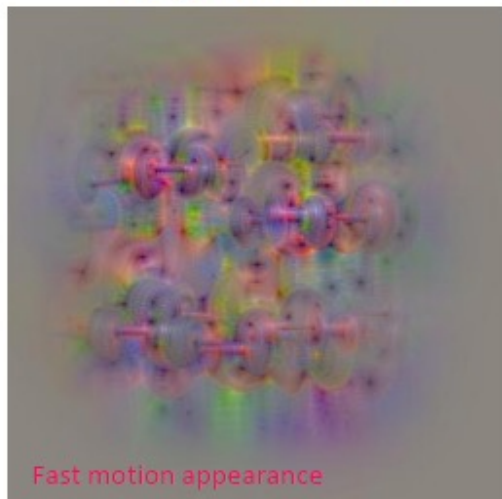
Figure credit: Simonyan and Zisserman, "Two-stream convolutional networks for action recognition in videos", NeurIPS 2014
 Feichtenhofer et al, "What have we learned from deep representations for action recognition?", CVPR 2018
 Feichtenhofer et al, "Deep insights into convolutional networks for video recognition?", IJCV 2019.

Slide credit: Justin Johnson

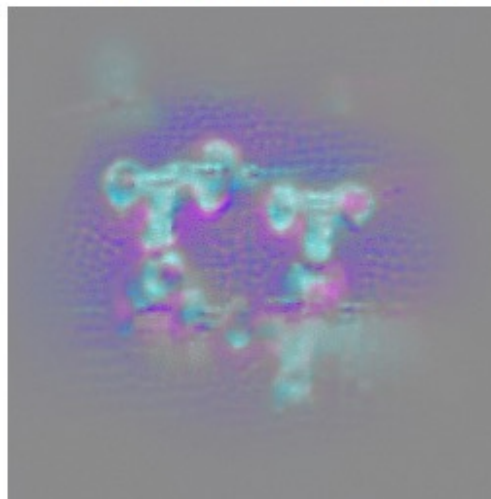
Visualization: Can you guess the action?

Feichtenhofer et al, "What have we learned from deep representations for action recognition?", CVPR 2018
Feichtenhofer et al, "Deep insights into convolutional networks for video recognition?", IJCV 2019.
Slide credit: Christoph Feichtenhofers

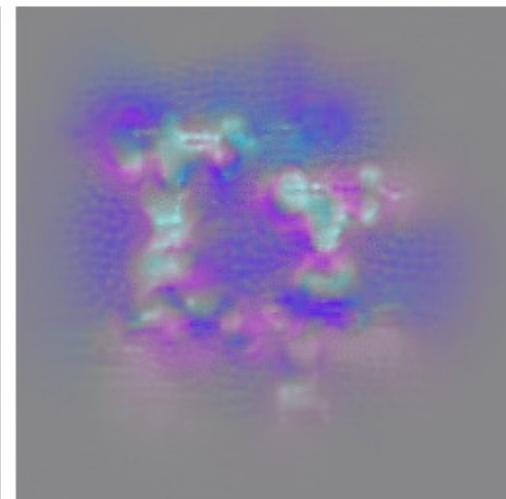
Appearance



"Slow" motion



"Fast" motion

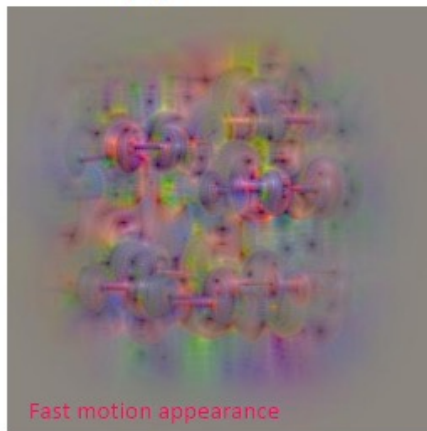


Slide credit: Justin Johnson

Visualization: Can you guess the action?

Weightlifting

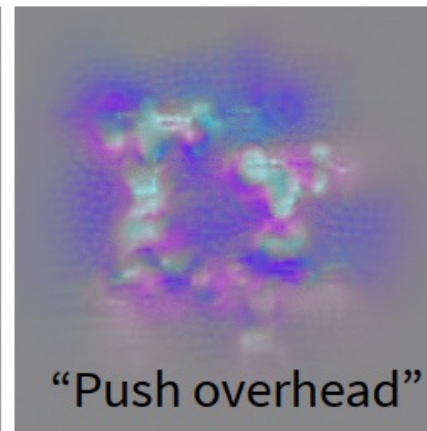
Appearance



“Slow” motion



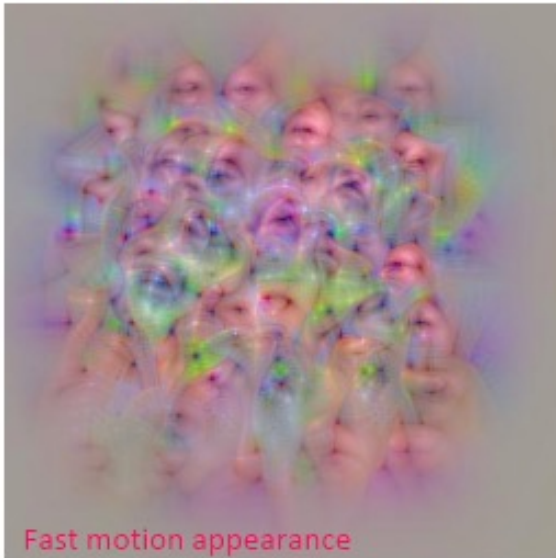
“Fast” motion



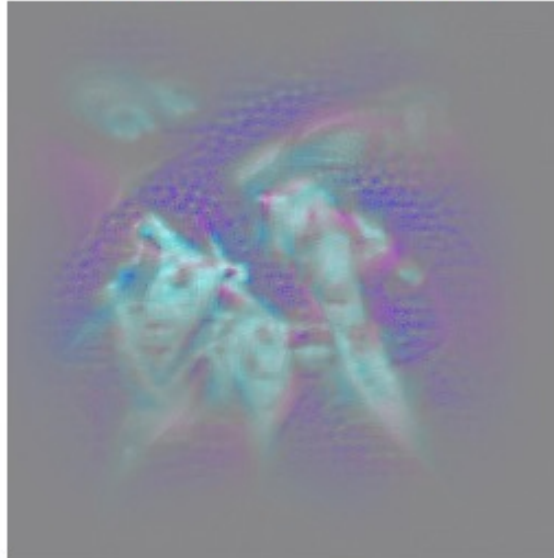
Slide credit: Justin Johnson

Visualization: Can you guess the action?

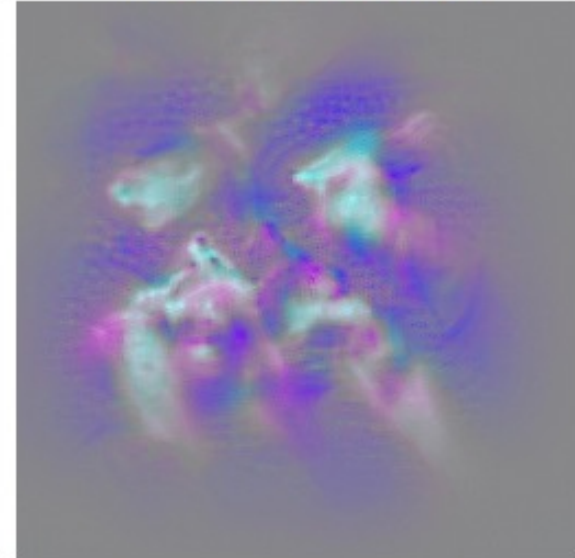
Appearance



“Slow” motion



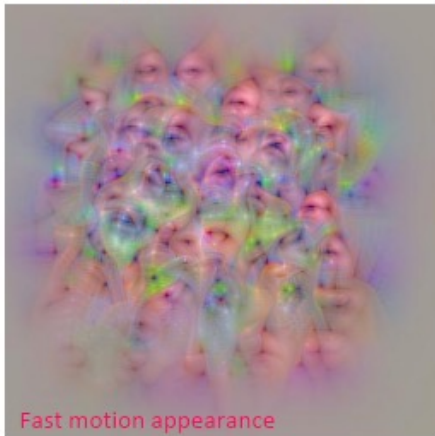
“Fast” motion



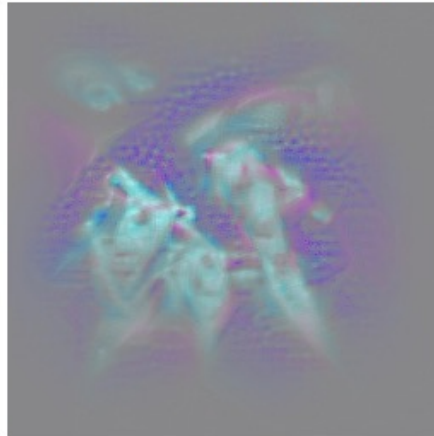
Visualization: Can you guess the action?

Apply Eye Makeup

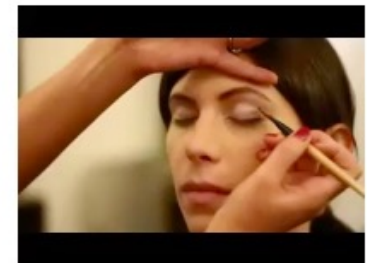
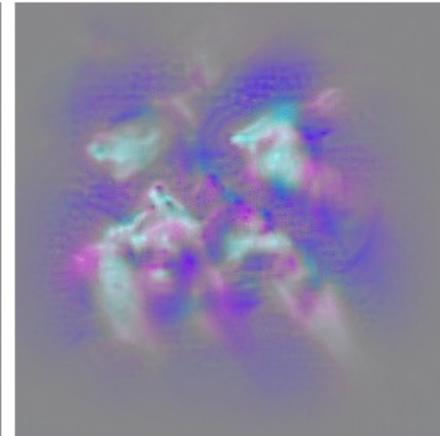
Appearance



“Slow” motion

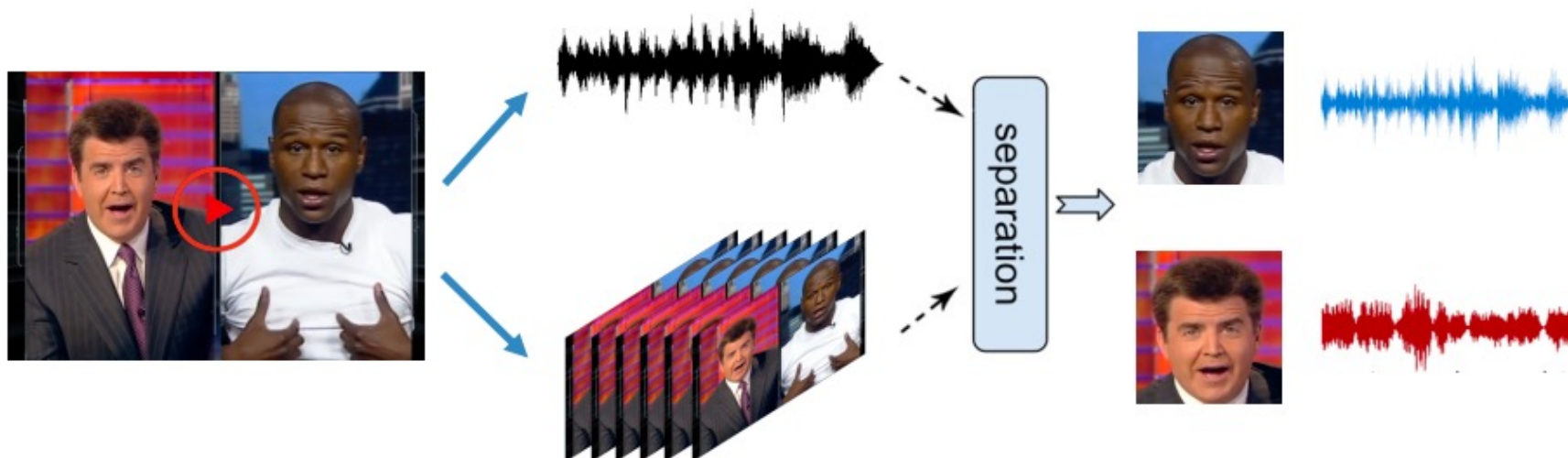


“Fast” motion



Slide credit: Justin Johnson

Frontiers: Visually-guided audio source separation



[Gao et al. ECCV 2018, Afouras et al. Interspeech'18, Gabby et al. Interspeech'18, Owens & Efros ECCV'18, Ephrat et al. SIGGRAPH'18, Zhao et al. ECCV 2018, Gao & Grauman ICCV 2019, Zhao et al. ICCV 2019, Xu et al. ICCV 2019, Gan et al. CVPR 2020, Gao et al. CVPR 2021]

Slide credit: Fei-Fei Li

Frontiers: Musical instruments source separation

Train on 100,000 unlabeled multi-source video clips,
then separate audio for novel video.



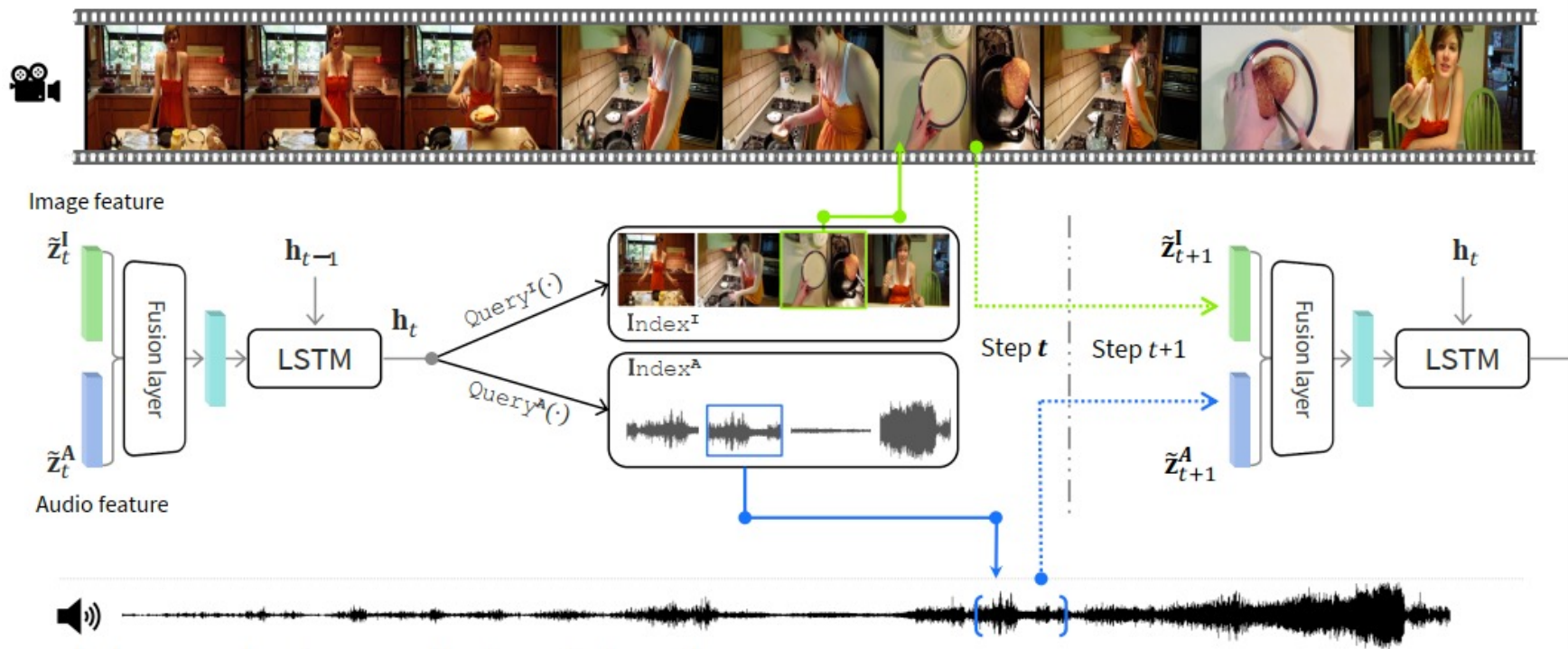
original video
(before separation)

object detections:
violin & flute

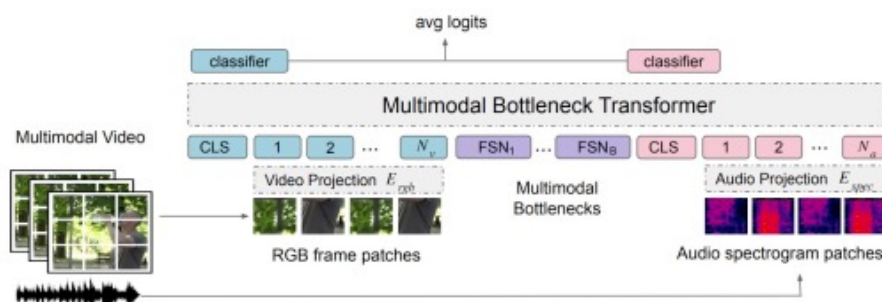
Gao & Grauman, Co-Separating Sounds of Visual Objects, ICCV 2019

Slide credit: Fei-Fei Li

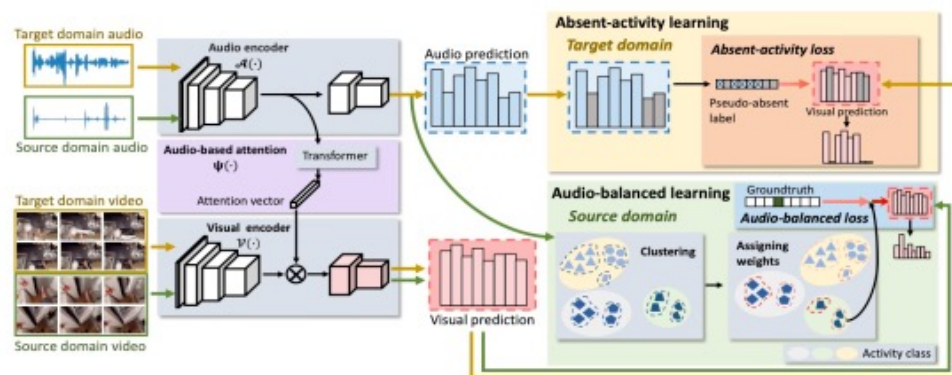
Frontiers: Audio as a preview mechanism for efficient action recognition in untrimmed videos



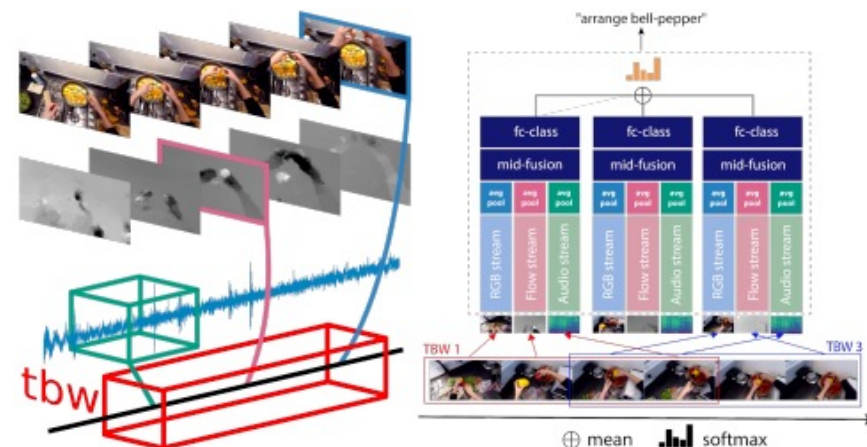
Frontiers: Multimodal Video Understanding



Attention Bottlenecks for Multimodal Fusion, Nagrani et al. NeurIPS 2021



Audio-Adaptive Activity Recognition Across Video Domains,
Yunhua et al. CVPR 2022



EPIC-Fusion: Audio-Visual Temporal Binding for Egocentric Action
Recognition, Kazakos et al., ICCV 2019

Tracking: some applications



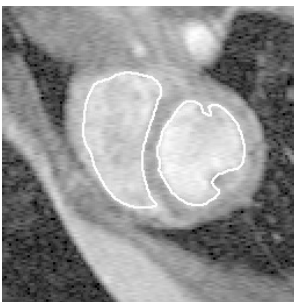
Body pose tracking,
activity recognition



Censusing a bat
population



Video-based
interfaces



Medical apps



Surveillance