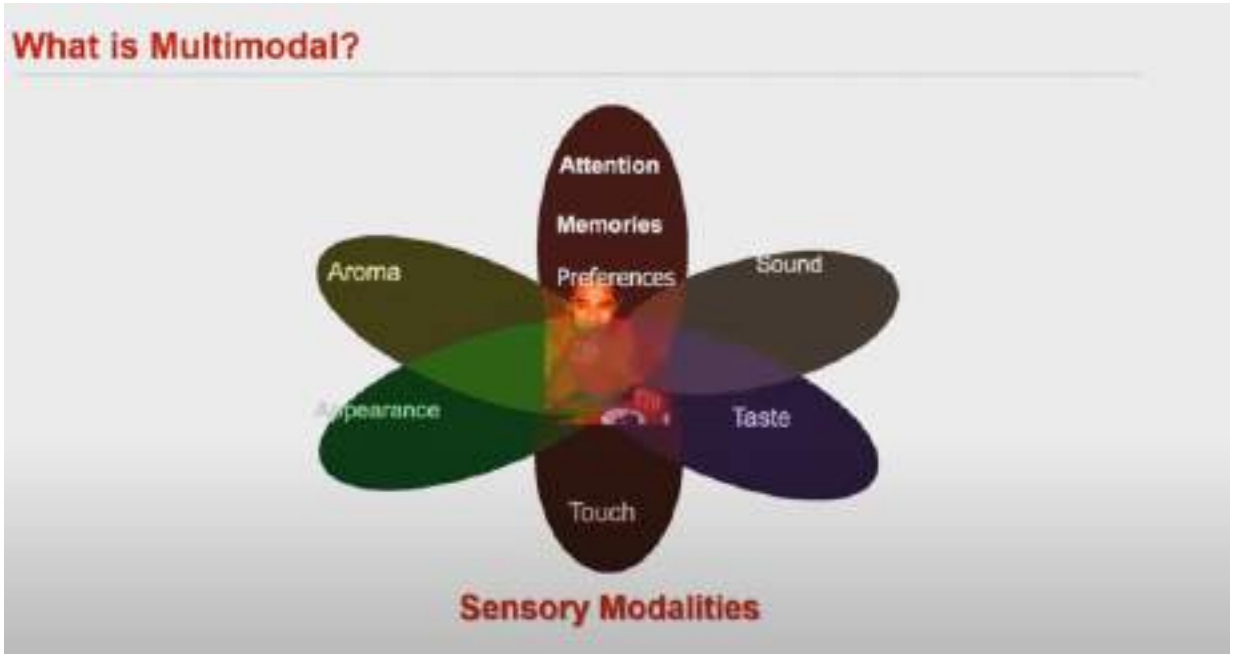
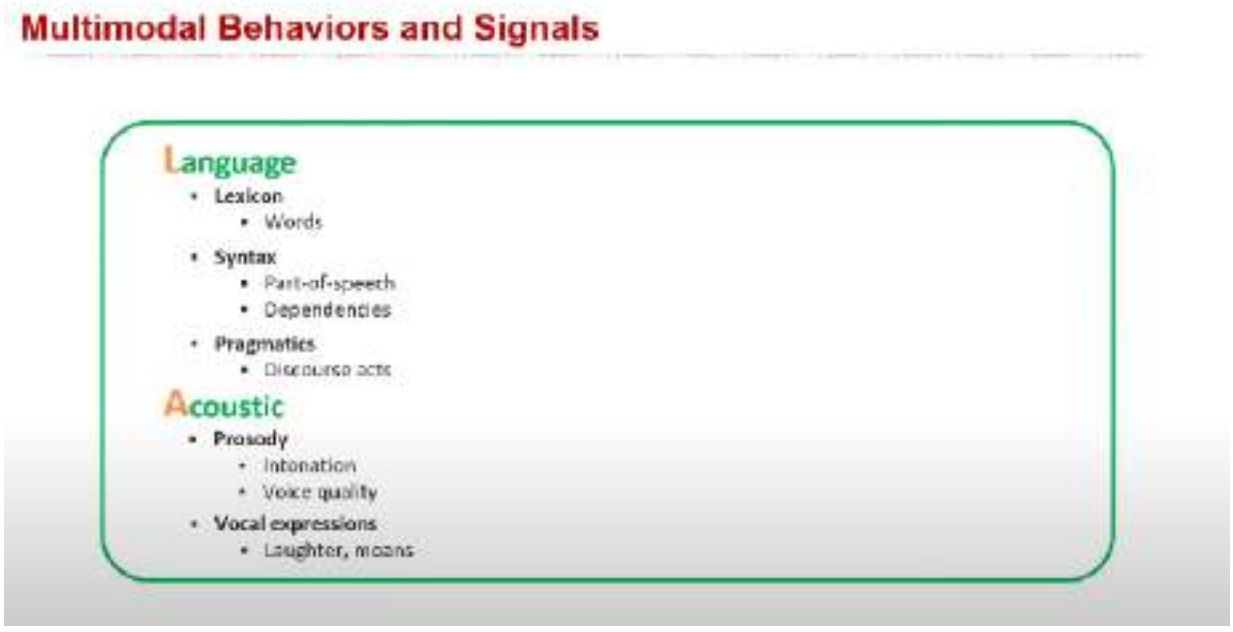
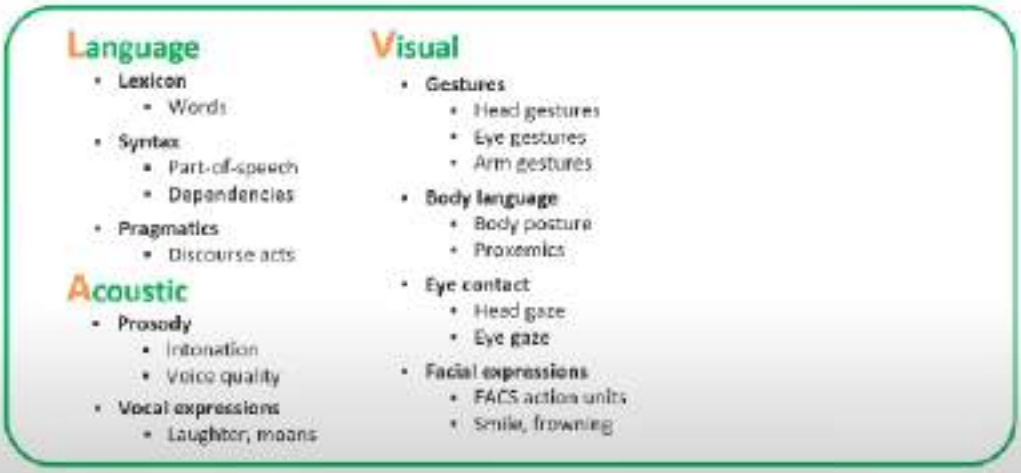
Multi Modal Lectures:





Multimodal Behaviors and Signals



What is Multimodal?

A dictionary definition…

Multimodal: with multiple modalities

A research-oriented definition…

***Multimodal* is the science of**

heterogeneous and interconnected data

Heterogeneous Modalities

Information present in different modalities will often show diverse qualities, structures and representations.

Abstract modalities are more likely to be homogeneous

Dimensions of Heterogeneity

Information present in different modalities will often show diverse qualities, structures, and representations.

*A **teacup** on the **right** of a **laptop** in a **clean room**.*

1. **Element representations:** discrete, continuous, granularity

{teacup, right, laptop, clean, room}

Dimensions of Heterogeneity

Information present in different modalities will often show diverse qualities, structures, and representations.

A teacup on the right of a laptop in a clean room.

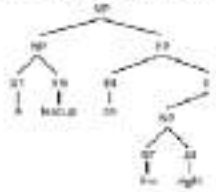
② **Element distributions:** density, frequency

▲ ▲ ▲ objects per image

●●●● words per minute

Dimensions of Heterogeneity

Information present in different modalities will often show diverse qualities, structures, and representations.



... Latent (implicit)

A teacup on the right ... Explicit (observable)

③ **Structure:** temporal, spatial, hierarchical, latent, explicit

Dimensions of Heterogeneity

Information present in different modalities will often show diverse qualities, structures, and representations.

A teacup on the right of a laptop in a clean room.

⑤ **Noise:** uncertainty, signal-to-noise ratio, missing data

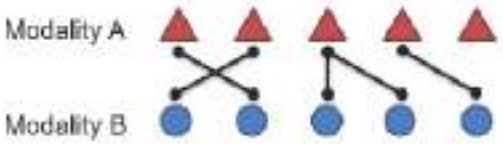
teacup → teacip

right → rihjt

Interconnected Modalities

① **Modality connections**

Modalities are often related and share commonality



② **Modality interactions**

Modality elements often interact during inference

Interactions happen during inference!

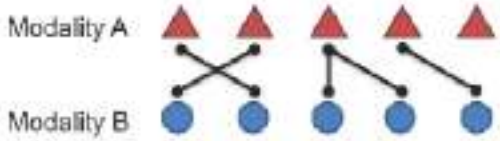
"Inference" examples:

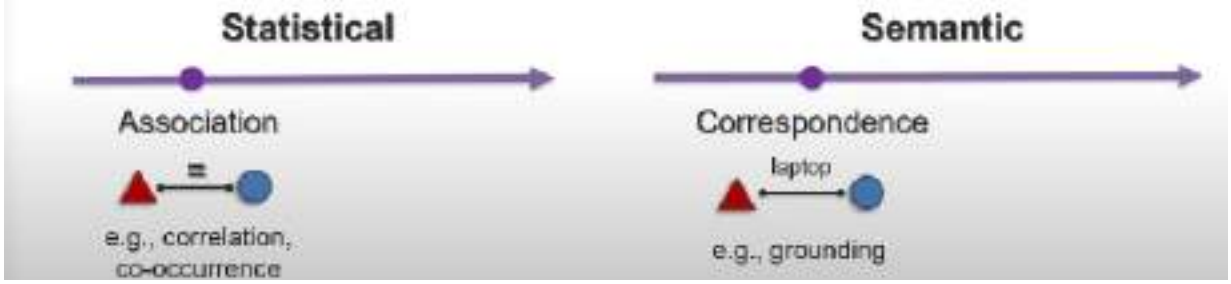
- Behavior perception
- Recognition task
- Modality translation

Interconnected Modalities

① **Modality connections**

Modalities are often related and share commonality

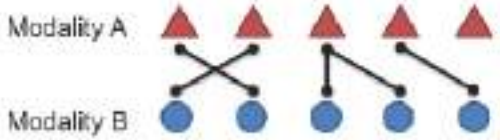


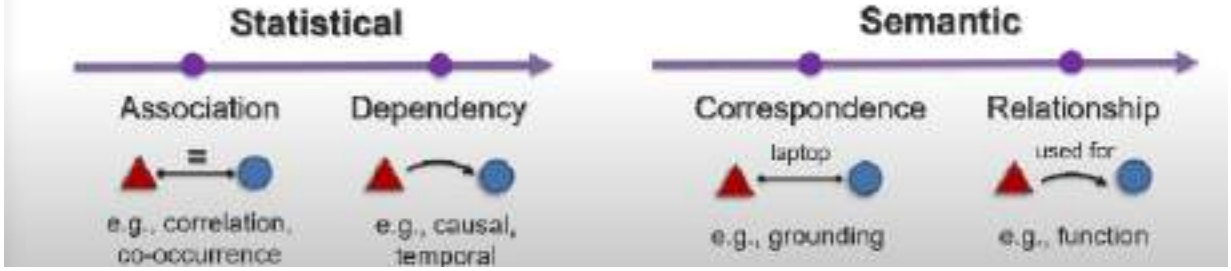


Interconnected Modalities

① **Modality connections**

Modalities are often related and share commonality





Interconnected Modalities

② **Modality interactions**

Modality elements often interact during inference

Is this indoors?

inference *Yes!*

A teacup on the right of a laptop in a clean room.

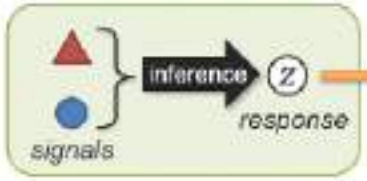
inference *Yes!*

Unimodal redundancy

Interconnected Modalities

② **Modality interactions**

Modality elements often interact during inference

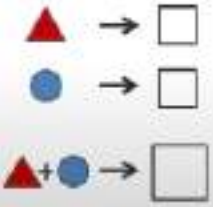


Is this indoors ?

A teacup on the right of a laptop in a clean room.

inference ***Yes!***

Unimodal redundancy

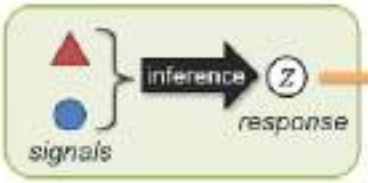


Multimodal enhancement

Interconnected Modalities

② **Modality interactions**

Modality elements often interact during inference



Types of interaction responses?

Is this a living room?

inference *Yes!*

A teacup on the right of a laptop in a clean room.

inference *No, probably study room.*

Taxonomy of Interaction Responses – A Behavioral Science View

	signal	response	signal	response	
Redundancy	a →	□	a+b →	□	Equivalence
	b →	□	a+b →	□	Enhancement
Nonredundancy	a →	□	a+b →	□ and ○	Independence
	b →	○	a+b →	□	Dominance
			a+b →	□ (or □)	Modulation

Multimodal Communication

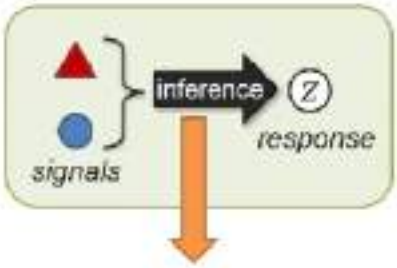
Taxonomy of Interaction Responses – A Behavioral Science View

	signal	response	signal	response	
Redundancy	a →	□	a+b →	□	Equivalence
	b →	□	a+b →	▭	Enhancement
Nonredundancy	a →	□	a+b →	□ and ○	Independence
	b →	○	a+b →	□	Dominance
			a+b →	▭ (or □)	Modulation
			a+b →	△	Emergence

Multimodal Communication

Dimensions of Modality Interactions

What are the dimensions for **digitally-represented** modalities?

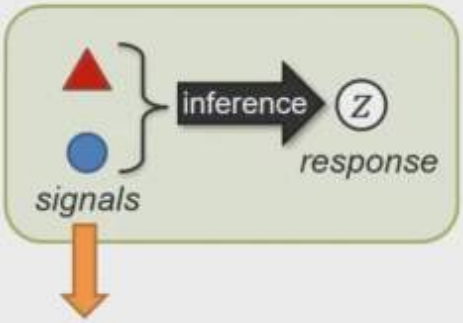


② Interaction Mechanics:

- Additive
- multiplicative
- Nonlinear
- Causal,
- Logical, …

Dimensions of Modality Interactions

What are the dimensions for **digitally-represented** modalities?



3 Input modalities:

- Unimodal
- Bimodal
- Trimodal
- High-modal, ...

Dimensions of Modality Interactions

What are the dimensions for **digitally-represented** modalities?

4 Context:

- Structure context
- Task relevance
- Context dependence
- High-modal, ...

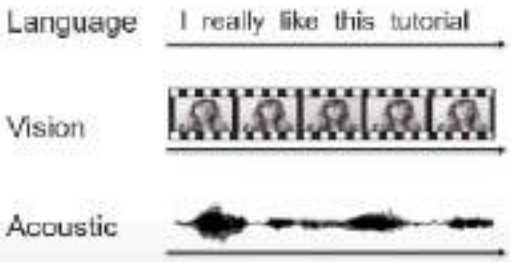
What is Multimodal Machine Learning?

Multimodal Machine Learning (ML) is the study of computer algorithms that learn and improve through the use and experience of data from multiple modalities

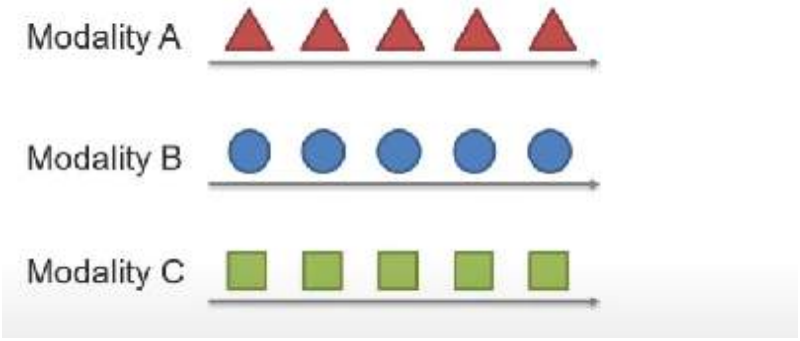
Multimodal Artificial Intelligence (AI) studies computer agents able to demonstrate intelligence capabilities such as understanding, reasoning and planning, through multimodal experiences, and data

Multimodal AI is a superset of Multimodal ML

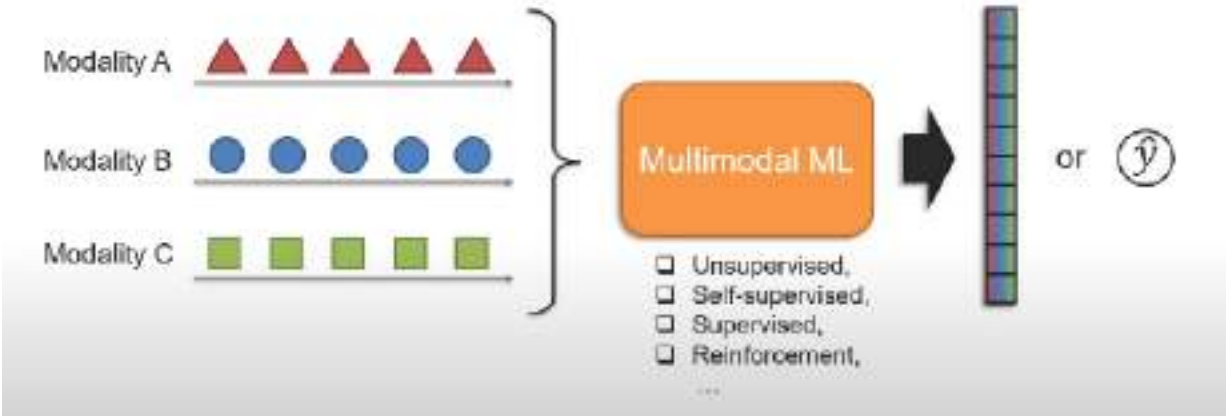
Multimodal Machine Learning



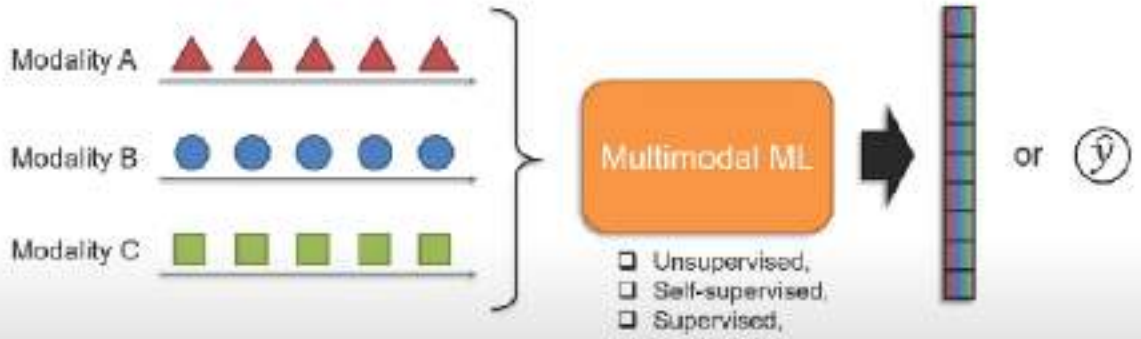
Multimodal Machine Learning



Multimodal Machine Learning



Multimodal Machine Learning



Challenge 1: Representation

Definition: Learning representations that reflect cross-modal interactions between individual elements, across different modalities

⟹ This is a core building block for most multimodal modeling problems!

Individual elements:

It can be seen as a "local" representation

or

representation using holistic features

Challenge 1: Representation

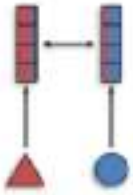
Definition: Learning representations that reflect cross-modal interactions between individual elements, across different modalities

Sub-challenges:

Fusion

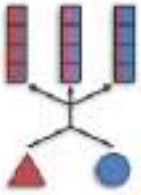
modalities > # representations

Coordination



modalities = # representations

Fission



modalities < # representations

Challenge 1: Representation

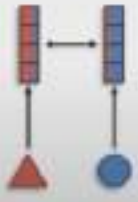
Definition: Learning representations that reflect cross-modal interactions between individual elements, across different modalities

Sub-challenges:

Fusion

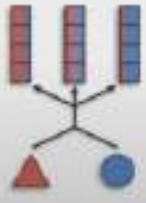
modalities > # representations

Coordination



modalities = # representations

Fission



modalities < # representations

Challenge 2: Alignment

Definition: Identifying and modeling cross-modal connections between all elements of multiple modalities, building from the data structure

➥ Most modalities have internal structure with multiple elements

Elements with temporal structure:

Other structured examples:

Spatial

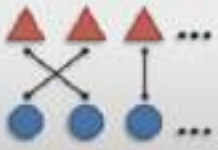
Hierarchical

Challenge 2: Alignment

Definition: Identifying and modeling cross-modal connections between all elements of multiple modalities, building from the data structure

Sub-challenges:

Connections



Explicit alignment
(e.g., grounding)

Aligned Representation

Alignment + representation
(aka, contextualized representation)

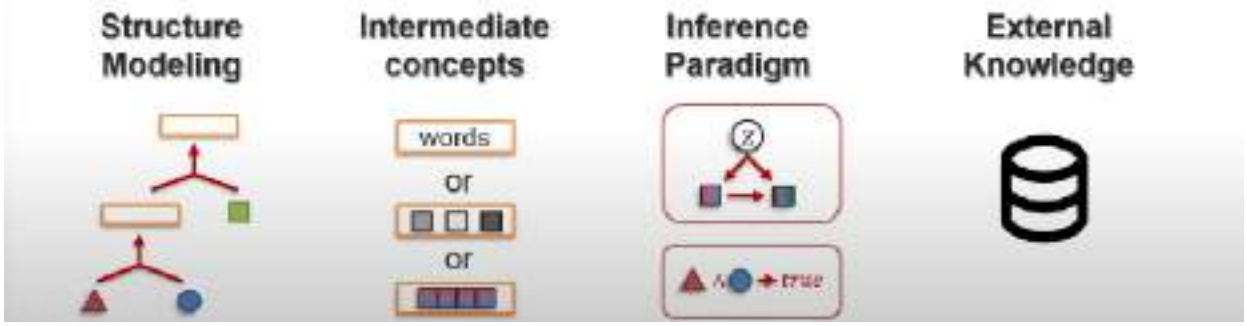
Elements

Segmentation of individual elements

Challenge 3: Reasoning

Definition: Combining knowledge, usually through multiple inferential steps, exploiting multimodal alignment and problem structure

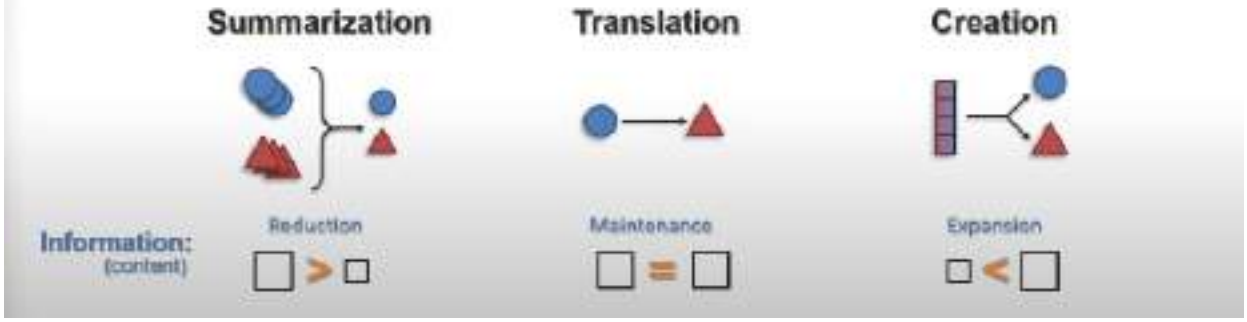
Sub-challenges:



Challenge 4: Generation

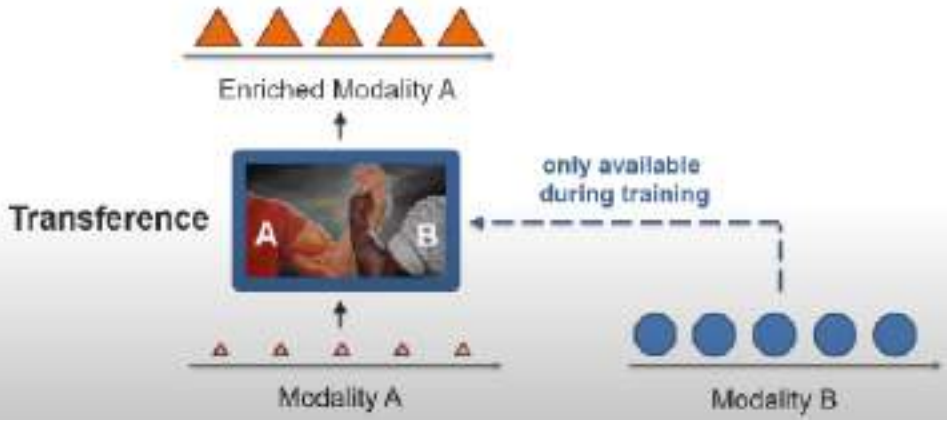
Definition: Learning a generative process to produce raw modalities that reflects cross-modal interactions, structure and coherence

Sub-challenges:



Challenge 5: Transference

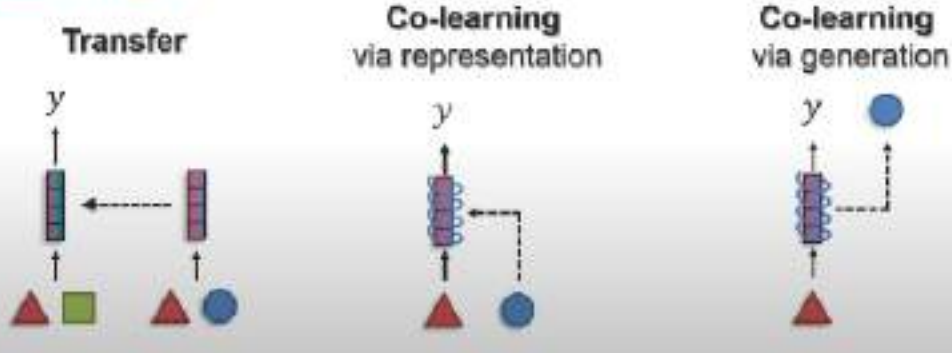
Definition: Transfer knowledge between modalities, usually to help the target modality which may be noisy or with limited resources



Challenge 5: Transference

Definition: Transfer knowledge between modalities, usually to help the target modality which may be noisy or with limited resources

Sub-challenges:



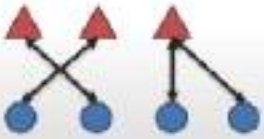
Challenge 6: Quantification

Definition: Empirical and theoretical study to better understand heterogeneity, cross-modal interactions and the multimodal learning process

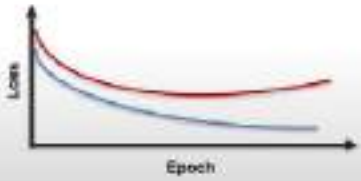
Sub-challenges:

Heterogeneity

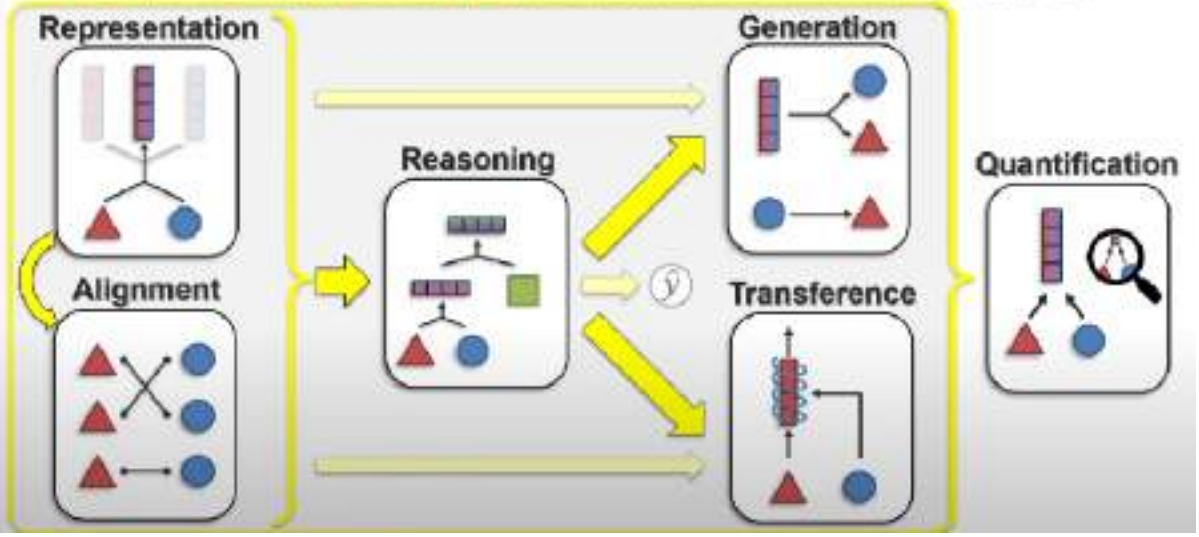
Interactions



Learning

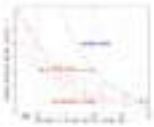


Core Multimodal Challenges



➢ The "Computational" Era(Late 1980s until 2000)

1) Audio-Visual Speech Recognition



Redundancy between audio and visual modalities help with handling noise and with robustness

2) Multimodal interfaces

***Affective Computing** is computing that relates to, arises from, or deliberately influences emotion or other affective phenomena.*

3) Multimedia

[1994-2010]

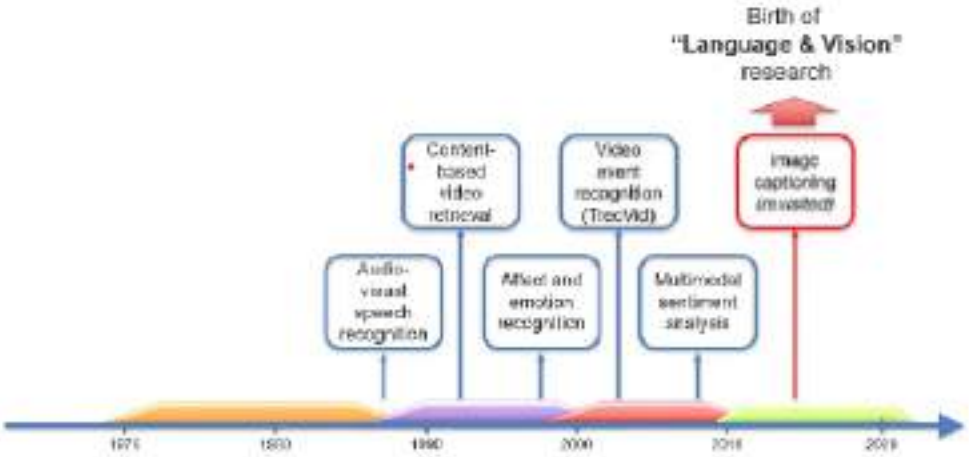
"...automatically combines speech, image and natural language understanding to create a full-content searchable digital video library."

➢ The "deep learning" era (2010s until ...)

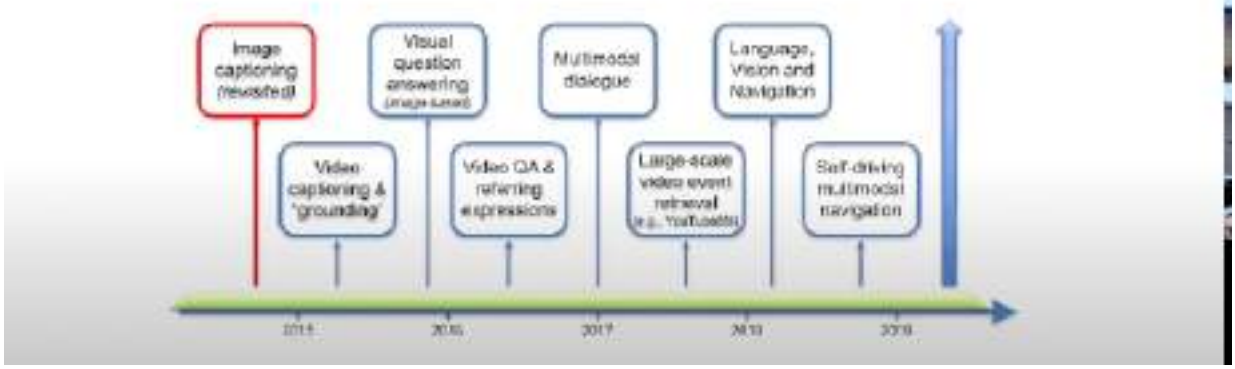
Representation learning (a.k.a. deep learning)

- Multimodal deep learning [ICML 2011]
- Multimodal Learning with Deep Boltzmann Machines [NIPS 2012]
- Visual attention: Show, Attend and Tell: Neural Image Caption Generation with Visual Attention [ICML 2015]

Multimodal Research Tasks



Multimodal Research Tasks



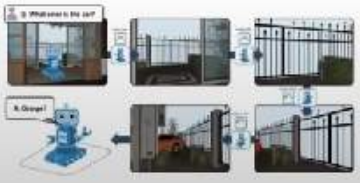
Real world tasks tackled by Multimodal ML

A. Affect recognition
   - Emotion
   - Personalities
   - Sentiment

B. Media description
   - Image and video captioning

C. Multimodal QA
   - Image and video QA
   - Visual reasoning

D. Multimodal Navigation
   - Language guided navigation
   - Autonomous driving



Real world tasks tackled by Multimodal ML

E. Multimodal Dialog
   - Grounded dialog

F. Event recognition
   - Action recognition
   - Segmentation

G. Multimedia information retrieval
   - Content based/Cross-media

Media description dataset 1 – MS COCO (B1)

- Microsoft Common Objects in COntext (MS COCO)
- 120000 images
- Each image is accompanied with five free form sentences describing it (at least 8 words)
- Sentences collected using crowdsourcing (Mechanical Turk)
- Also contains object detections, boundaries and keypoints

Visual Questions & Answers – VQA (C1)

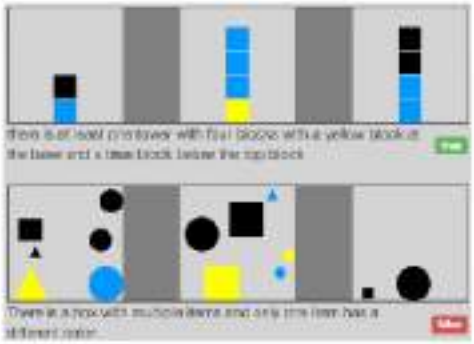
- Task - Given an image and a question, answer the question (http://www.visualqa.org/)

Social Interaction Q&A Dataset (A10)

- Social-IQ: 1.2k videos, 7.5k questions, 50k answers
- Questions and answers centered around social behaviors

Multimodal QA – Visual Reasoning (C9)

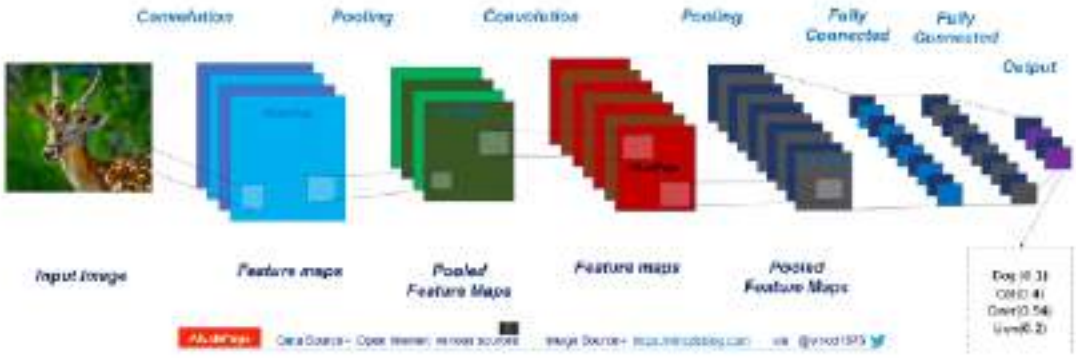
- Cornell NLVR
  - 92,244 pairs of natural language statements grounded in synthetic images
  - Determine whether a sentence is true or false about an image



Some Advice About Multimodal Datasets

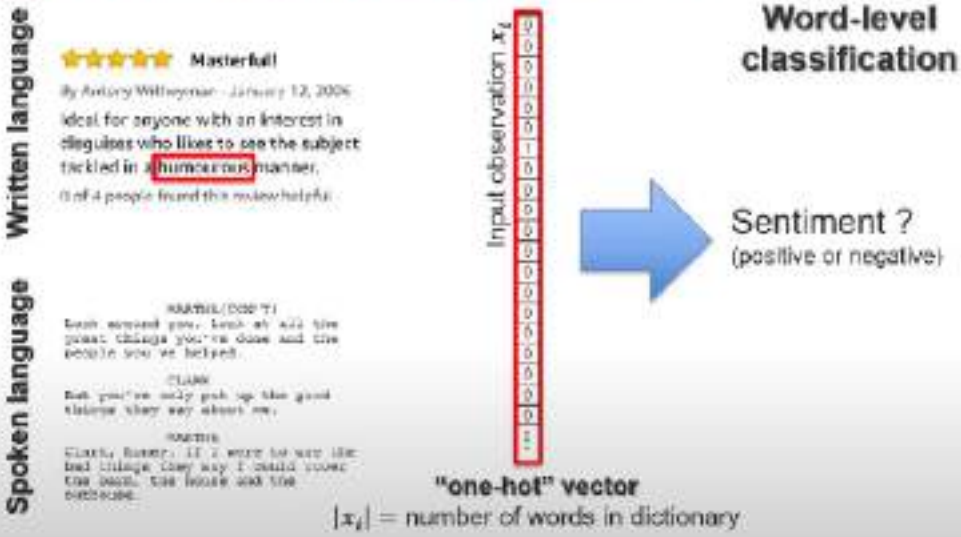
- If you are used to deal with text or speech
  - Space will become an issue working with image/video data
  - Some datasets are in 100s of GB (compressed)
- Memory for processing it will become an issue as well
  - Won't be able to store it all in memory
- Time to extract features and train algorithms will also become an issue
- Plan accordingly!
  - Sometimes tricky to experiment on a laptop (might need to do it on a subset of data)

Convolutional Neural Network

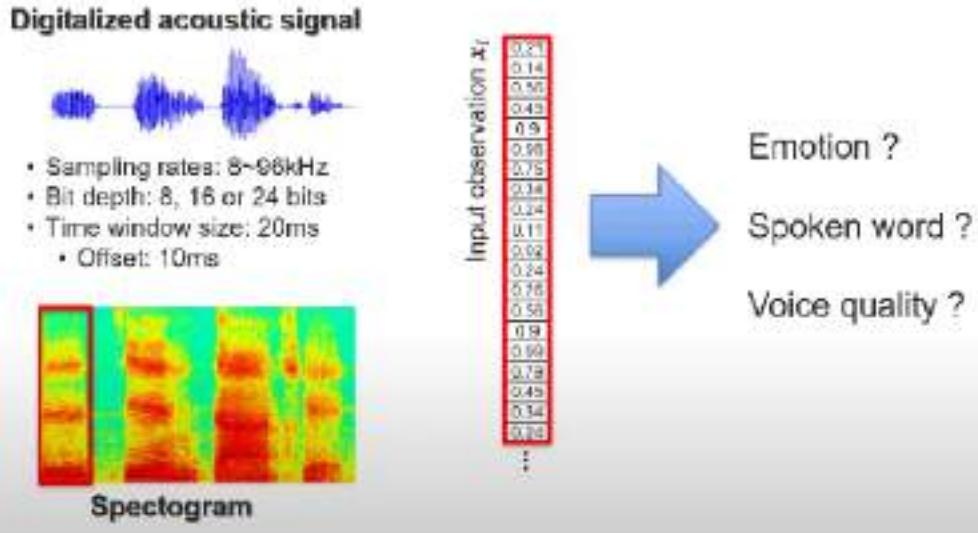


Lecture 2.1 - Basic Concepts

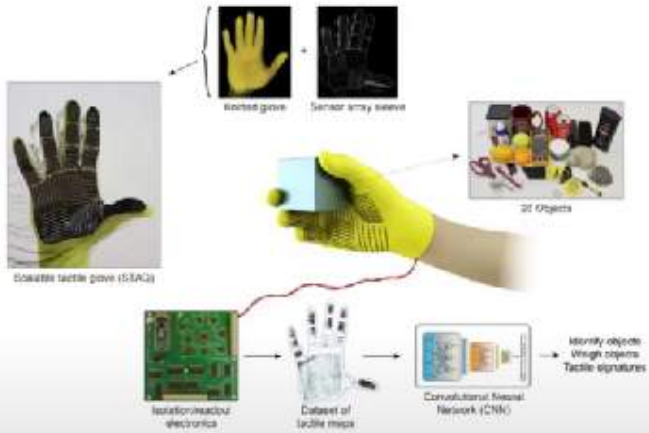
Unimodal Representation – Language Modality



Unimodal Representation – Acoustic Modality



Unimodal Representation – Sensors



The tactile sensor array (548 sensors) is assembled on a knitted glove uniformly distributed over the hand.

40 mins lect 2.1

Simple Classifier: Nearest Neighbor

Distance metrics

L1 (Manhattan) distance:

$$d_1(x_1, x_2) = \sum_j \left| x_1^j - x_2^j \right|$$

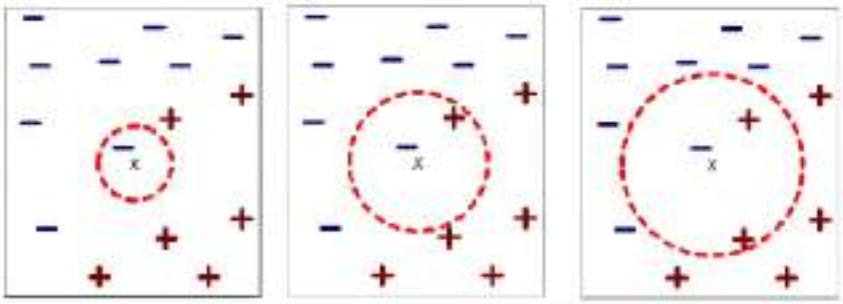
L2 (Eucledian) distance:

$$d_2(x_1, x_2) = \sqrt{\sum_j \left(x_1^j - x_2^j \right)^2}$$

Which distance metric to use?

First hyper-parameter!

Definition of K-Nearest Neighbor



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

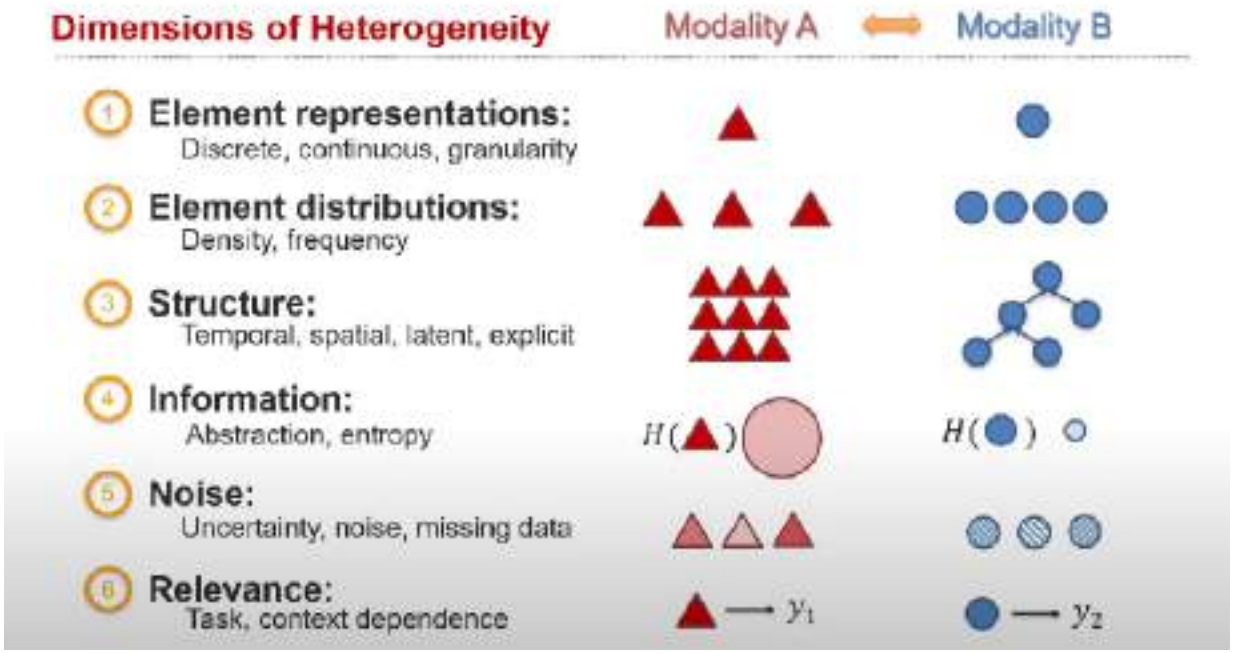
What value should we set K?

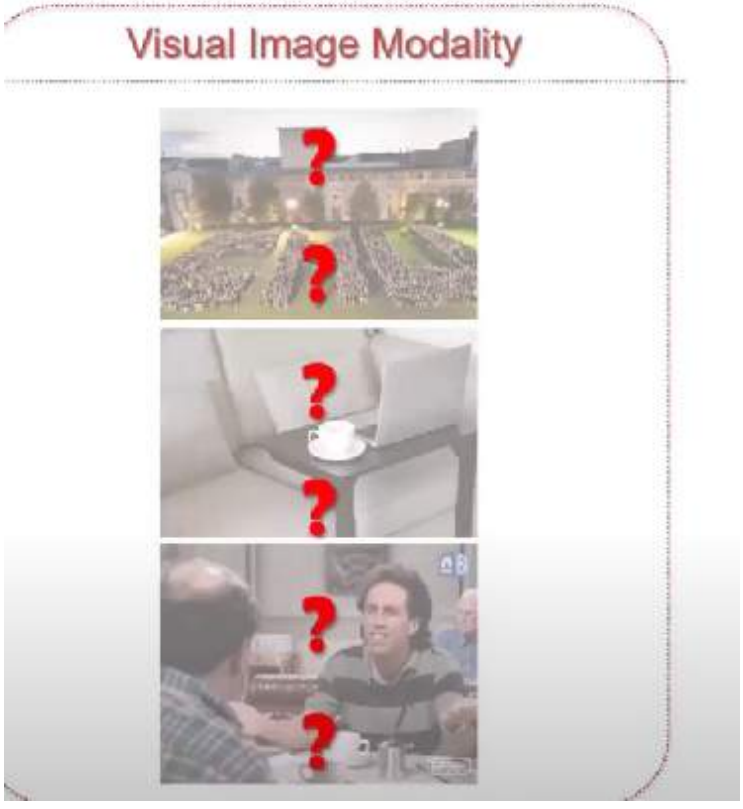
Second hyper-parameter!

Heterogeneous Modalities

Information present in different modalities will often show diverse qualities, structures and representations.

If Mod A is let us text and Mod B is speech , they are different because they have different structures in tensors or so on


Dimensions of Heterogeneity
Modality A
Modality B
1 Element representations:
Discrete, continuous, granularity
2 Element distributions:
Density, frequency
3 Structure:
Temporal, spatial, latent, explicit
4 Information:
Abstraction, entropy
H( )
H( )
5 Noise:
Uncertainty, noise, missing data
6 Relevance:
Task, context dependence
y_1
y_2


Visual Image Modality

Objects in an image also have attributes associated with them as shown below :

Object-Based Visual Representation

Object-Based Visual Representation

Object Descriptors

How to represent and detect an object?

Many approaches over the years…

Image gradient

Edge detection

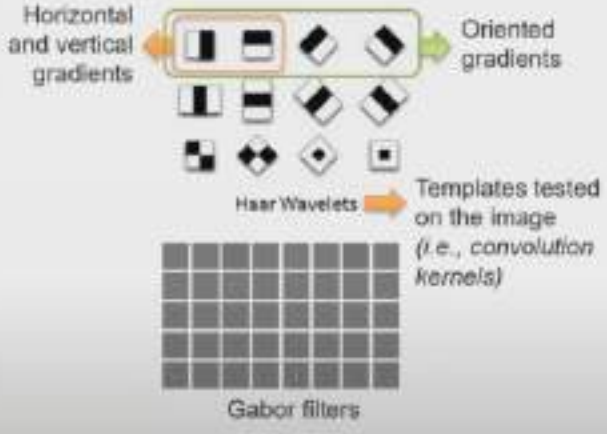
Histograms of Oriented Gradients

Optical Flow

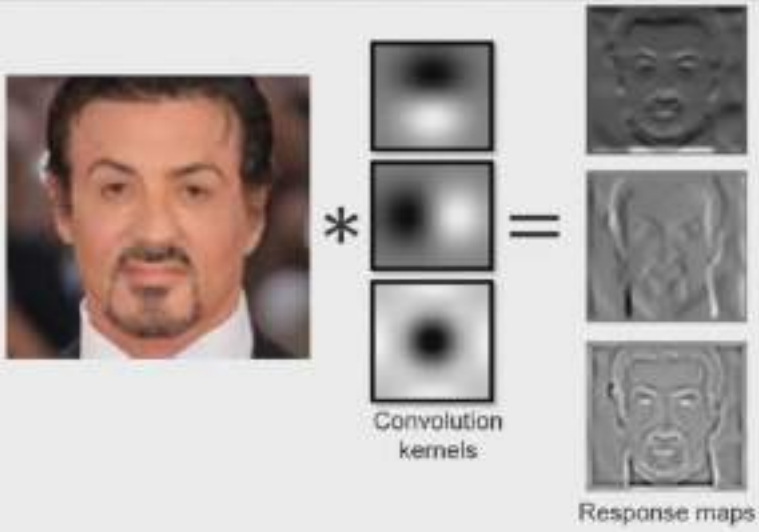
Object Descriptors

How to represent and detect an object?

Many approaches over the years…



Convolution Kernels

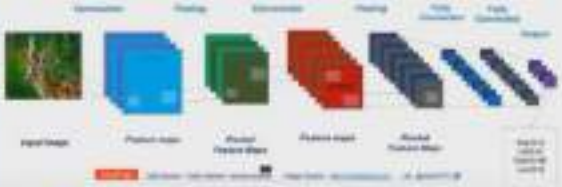


Object Descriptors

How to represent and detect an object?

Many approaches over the years…

Convolutional Neural Network (CNN)



More details about CNNs is coming…

Why using Convolutional Neural Networks?

Goal: building more abstract, hierarchical visual representations

Objects

Parts

Edges/blobs

Input pixels

Key advantages:

1) Inspired from visual cortex
2) Encourages visual abstraction
3) Exploits *translation invariance*
4) Kernels/templates are learned
5) Fewer parameters than MLP

Translation Invariance

2 Data Points – Which one is up?

- MLP can easily learn this task (possibly with only 1 neuron!)

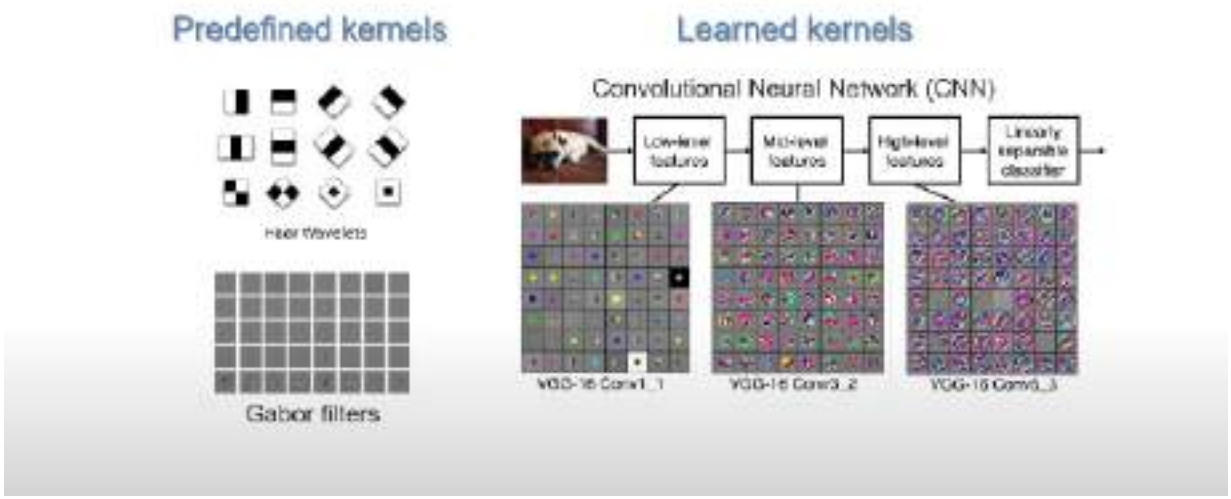
What happens if the face is slightly translated?

- The model should still be able to classify it

Conventional MLP models are not translation invariant!

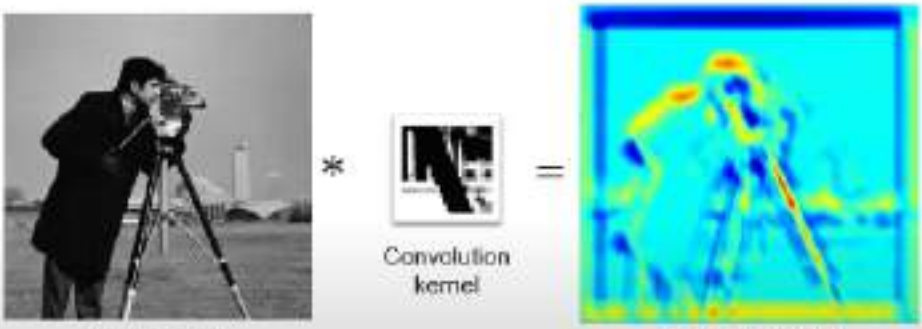
- But CNNs are kernel-based, which helps with translation invariance and reduce number of parameters

Predefined vs Learned Kernels

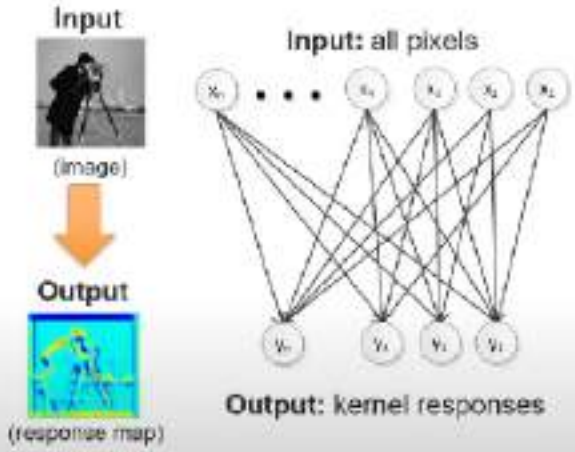


Learned Filters (aka Convolution Kernels)

Convolution in 2D – Example



Convolution as a Fully-Connected Network



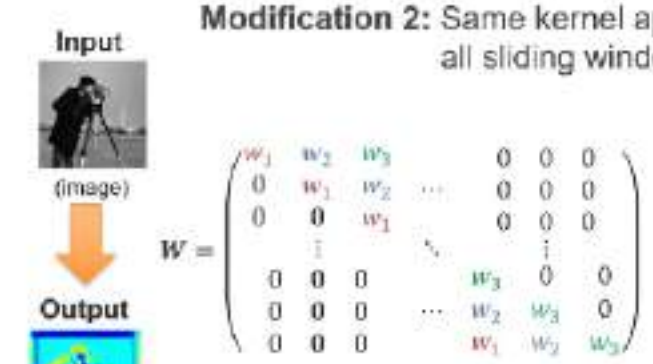
Not efficient!

200×200 image requires $40{,}000 \times n$ parameters

(where n is size of kernel)

Convolutional Neural Layer

Modification 2: Same kernel applied to all sliding windows

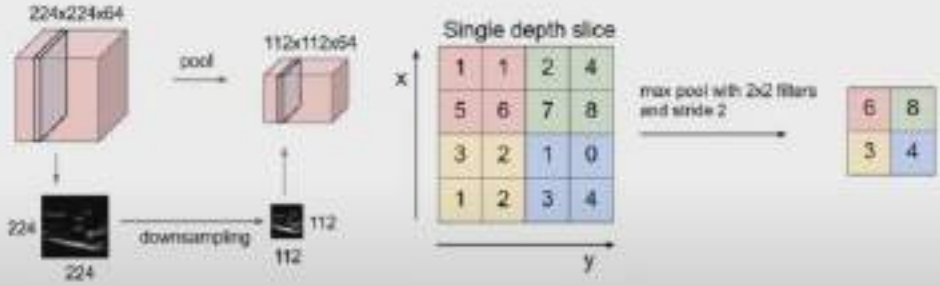


- Can be implemented efficiently on GPUs
- W will be 3D: 3rd dimension allows for multiple kernels

Pooling Layer

Response map subsampling:

Allows summarization of the responses



Common architectures

Repeat several times:

- Start with a convolutional layer
- Followed by non-linear activation and pooling

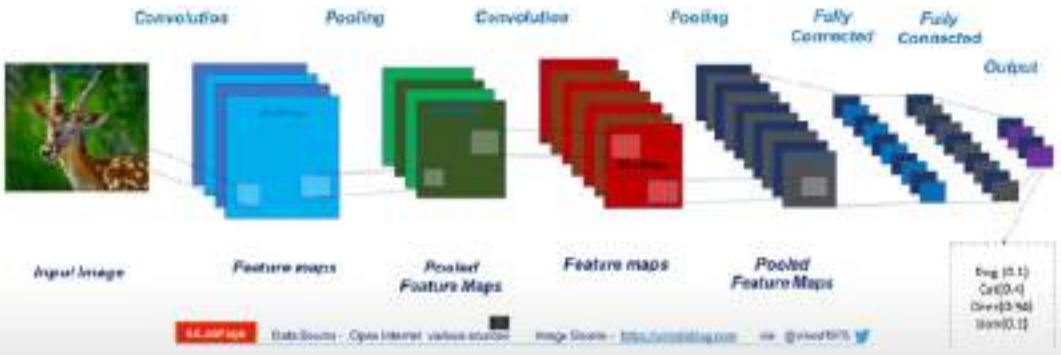
End with a fully connected (MLP) layer

Visualizing the Last CNN Layer: t-sne

Alex Net

Embed high dimensional data points (i.e. feature codes) so that pairwise distances are conserved in local neighborhoods.

Convolutional Neural Network



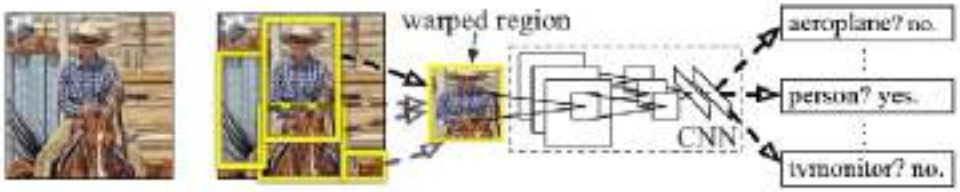
ImageNet

- Objects already centered, ready for training
- Hierarchy similar to FrameNet (originally designed for words)

Object Detection (and Segmentation)

A better option: Start by identifying hundreds of region proposals and then apply our CNN object detector

R-CNN [Girshick et al., CVPR 2014]



- Select ~2000 region proposals
- Warp each region
- Apply CNN to each region

Word Modalities:

Simple Word Representation

What is the meaning of "bardiwac"?

- He handed her her glass of bardiwac.
- Beef dishes are made to complement the bardiwacs.
- Nigel staggered to his feet, face flushed from too much bardiwac.
- Malbec, one of the lesser-known bardiwac grapes, responds well to Australia's sunshine.
- I dined off bread and cheese and this excellent bardiwac.
- The drinks were delicious: blood-red bardiwac as well as light, sweet Rhenish.

⇒ bardiwac is a heavy red alcoholic beverage made from grapes

How to learn (word) features/representations?

- **Distribution hypothesis**: Approximate the word meaning by its surrounding words
- Words used in a similar context will lie close together

How to learn (word) features/representations?

- **Distribution hypothesis**: Approximate the word meaning by its surrounding words
- Words used in a similar context will lie close together

- **Instead of capturing co-occurrence counts directly, predict surrounding words of every word**

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\le j\le c, j\ne 0}\log p(w_{t+j}|w_t)$$

Geometric interpretation

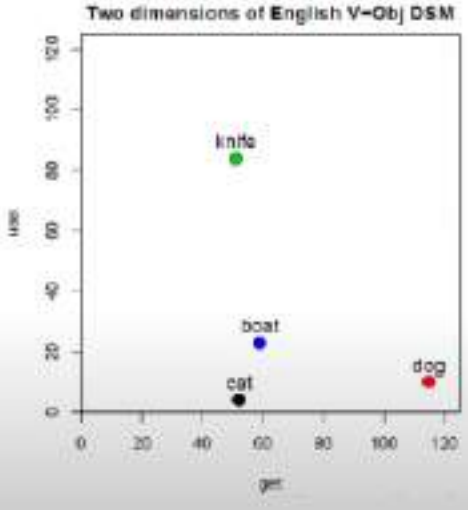
- row vector $\mathbf{x}_{dog}$ describes usage of word *dog* in the corpus
- can be seen as coordinates of point in *n*-dimensional Euclidean space R^n

	get	see	use	hear	eat	kill
knife	51	20	84	0	3	0
cat	52	58	4	4	6	26
dog	115	83	10	42	33	17
boat	59	39	23	4	0	0
cup	98	14	6	2	1	0
pig	12	17	3	2	9	27
banana	11	2	2	0	18	0

co-occurrence matrix **M**

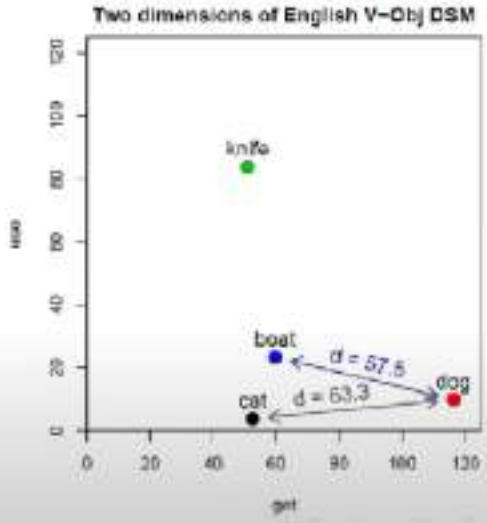
Distance and similarity

- illustrated for two dimensions: *get* and *use*: $\mathbf{x}_{dog}$ = (115, 10)
- similarity = spatial proximity (Euclidean distance)
- location depends on frequency of noun ($f_{dog} \approx 2.7 \cdot f_{cat}$)



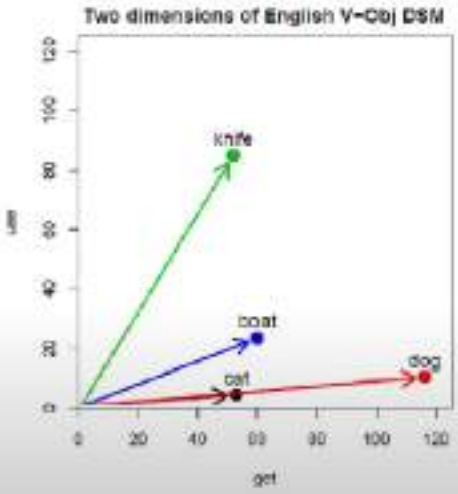
Distance and similarity

- illustrated for two dimensions: *get* and *use*: $\mathbf{x}_{dog}$ = (115, 10)
- similarity = spatial proximity (Euclidean distance)
- location depends on frequency of noun ($f_{dog} \approx 2.7 \cdot f_{cat}$)

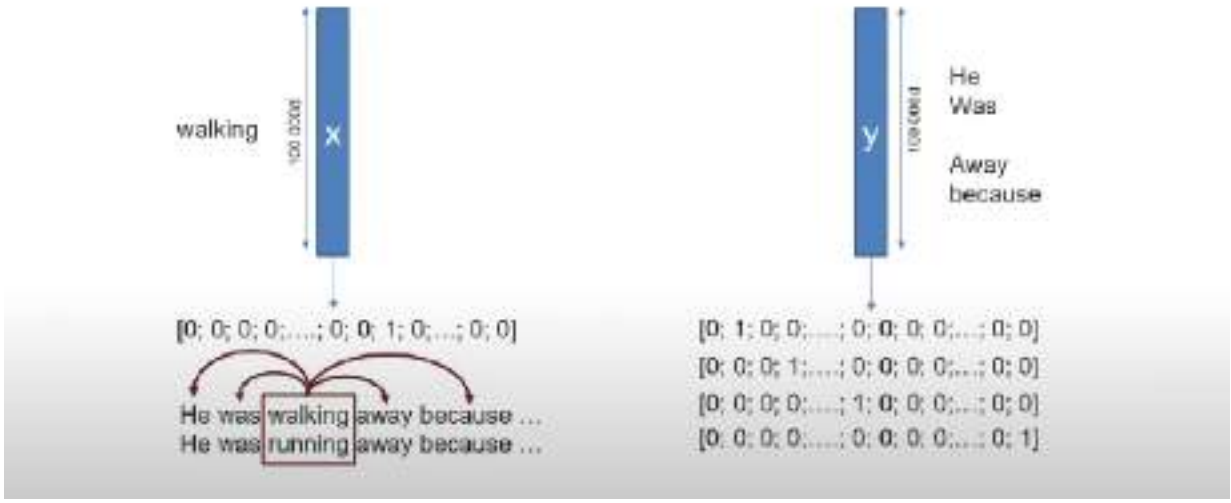


Angle and similarity

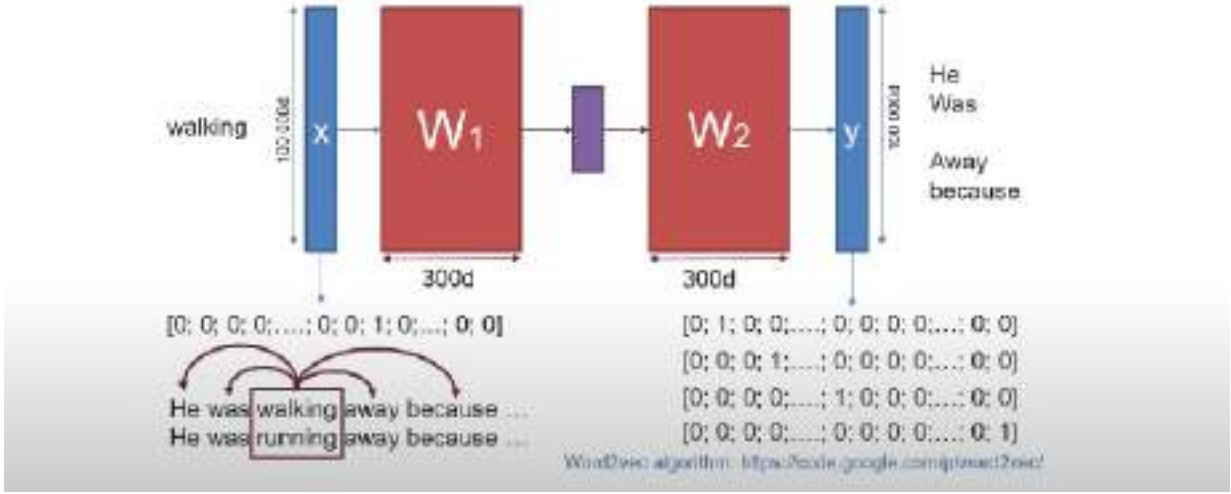
- direction more important than location
- normalise "length" $||\mathbf{x}_{dog}||$ of vector

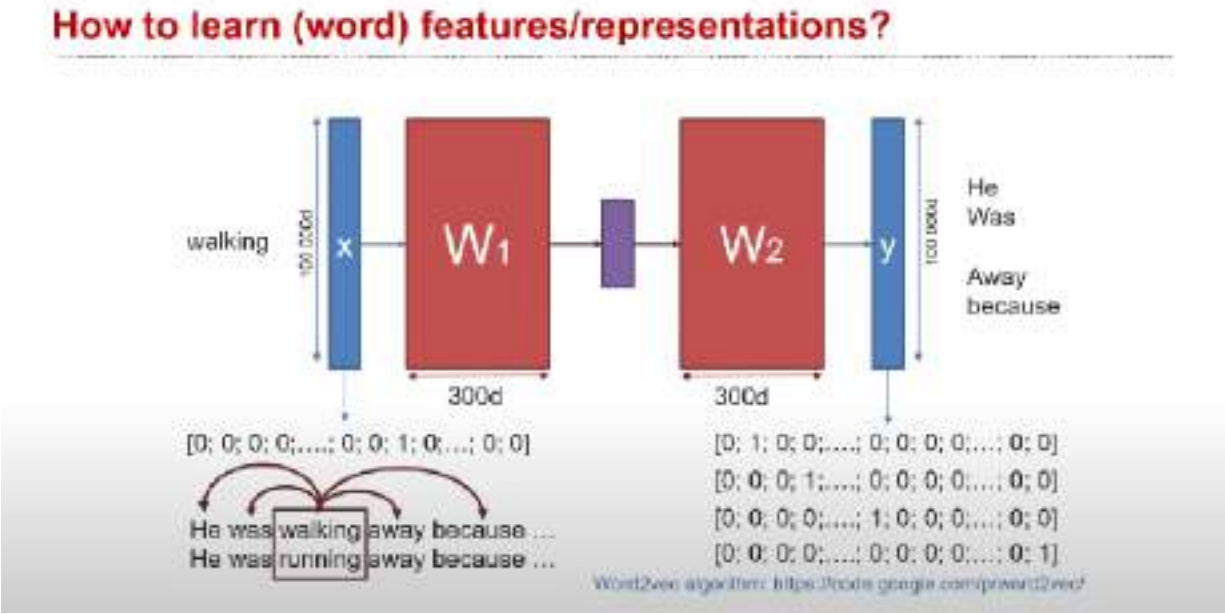


How to learn (word) features/representations?



How to learn (word) features/representations?

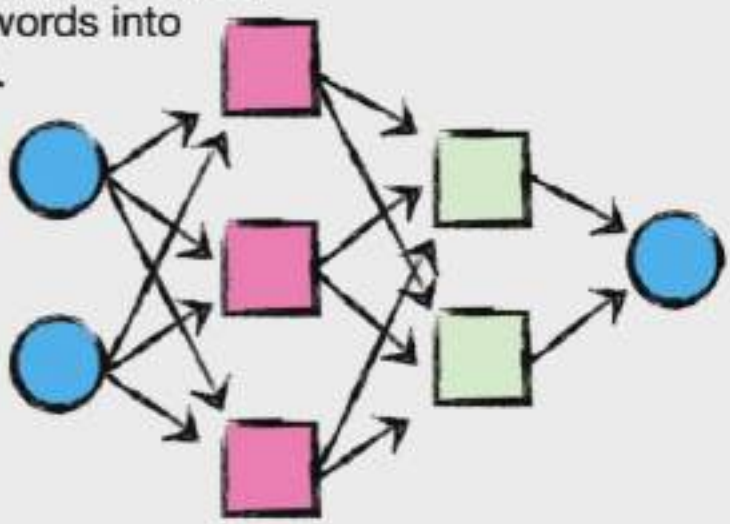




25 mins

Word2Vec Explained:

So, if we want to plug words into a **Neural Network**, or some other machine learning algorithm, we need a way to turn the words into numbers.



How to use these word representations

If we would have a vocabulary of 100 000 words:

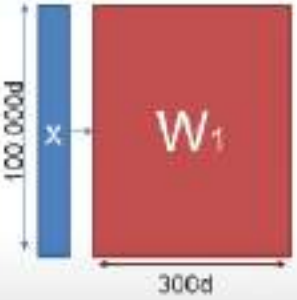
Classic NLP: 100 000 dimensional vector

Walking: [0; 0; 0; 0;….; 0; 0; 1; 0;…; 0; 0]

Running: [0; 0; 0; 0;….; 0; 0; 0; 0;…; 1; 0]

Similarity = 0.0

Transform: x'=x*W



Goal: 300 dimensional vector

Walking: [0,1; 0,0003; 0;….; 0,02; 0.08; 0.05]

Running: [0,1; 0,0004; 0;….; 0,01; 0.09; 0,05]

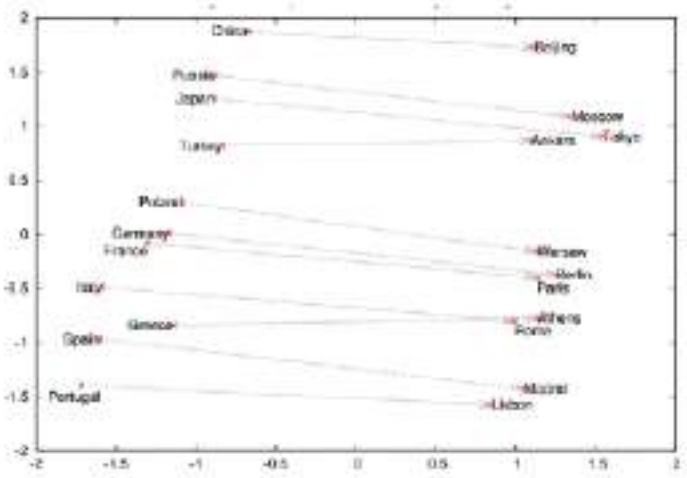
Similarity = 0.9

Vector space models of words

- While learning these word representations, we are actually building a vector space in which all words reside with certain relationships between them
- Encodes both syntactic and semantic relationships
- This vector space allows for algebraic operations:

Vec(king) – vec(man) + vec(woman) ≈ vec(queen)

Vector space models of words: semantic relationships



Trained on the Google news corpus with over 300 billion words

Lexicon-based Word Representation

LIWC: Language Inquiry & Word Count

Manually created dictionaries for different topics and categories:

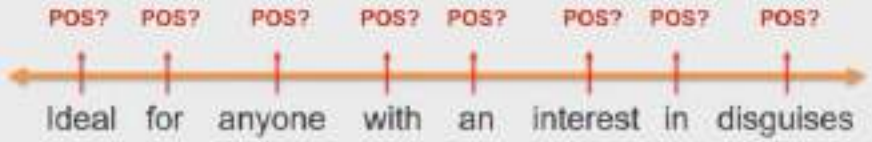
- Function words: *pronouns, preposition, negation…*
- Affect words: *positive, negative emotions*
- Social words: *family, friends, referents*
- Cognitive processes: *Insight, cause, …*
- Perceptual processes: *Seeing, hearing, feeling*
- Biological processes: *Body, health/illness,…*
- Drives and needs: *Affiliation, achievement, …*
- Time orientation: *past, present, future*
- Relativity: *motion, space, time*
- Personal concerns: *work, leisure, money, religion …*
- Informal speech: *swear words, fillers, assent,…*

LIWC can encode individual words or full sentences.

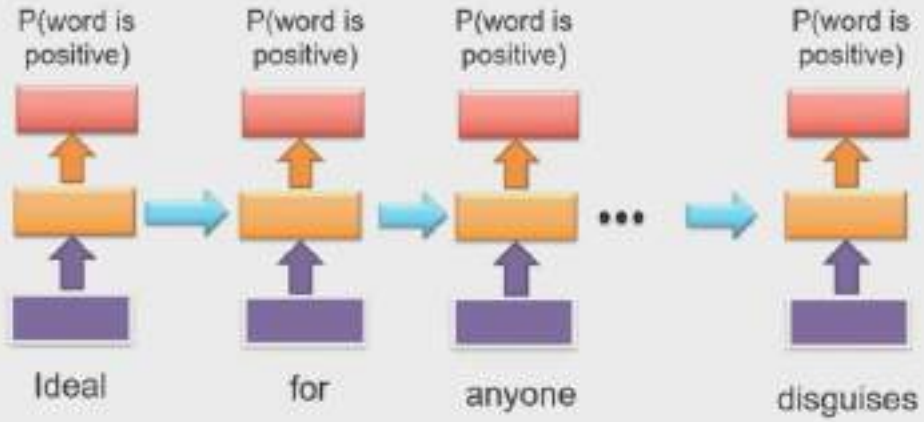
https://liwc.wpengine.com/

Commercial software. Contact TAs in advance if you would like to use it.

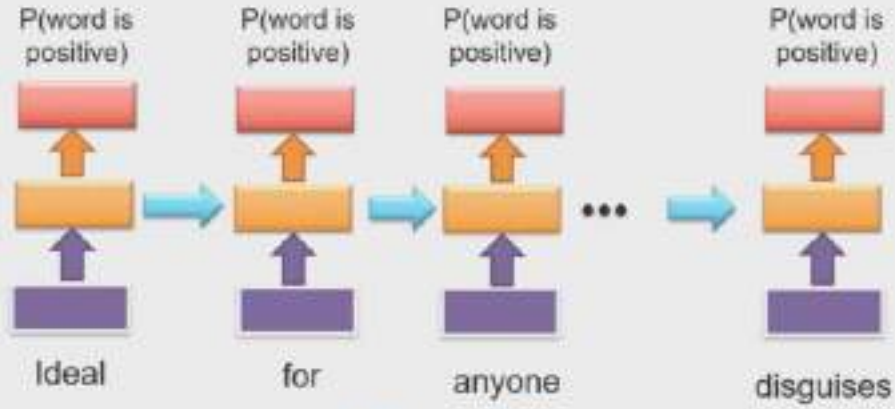
Sentence Modeling: Sequence Prediction



RNN for Sequence Prediction

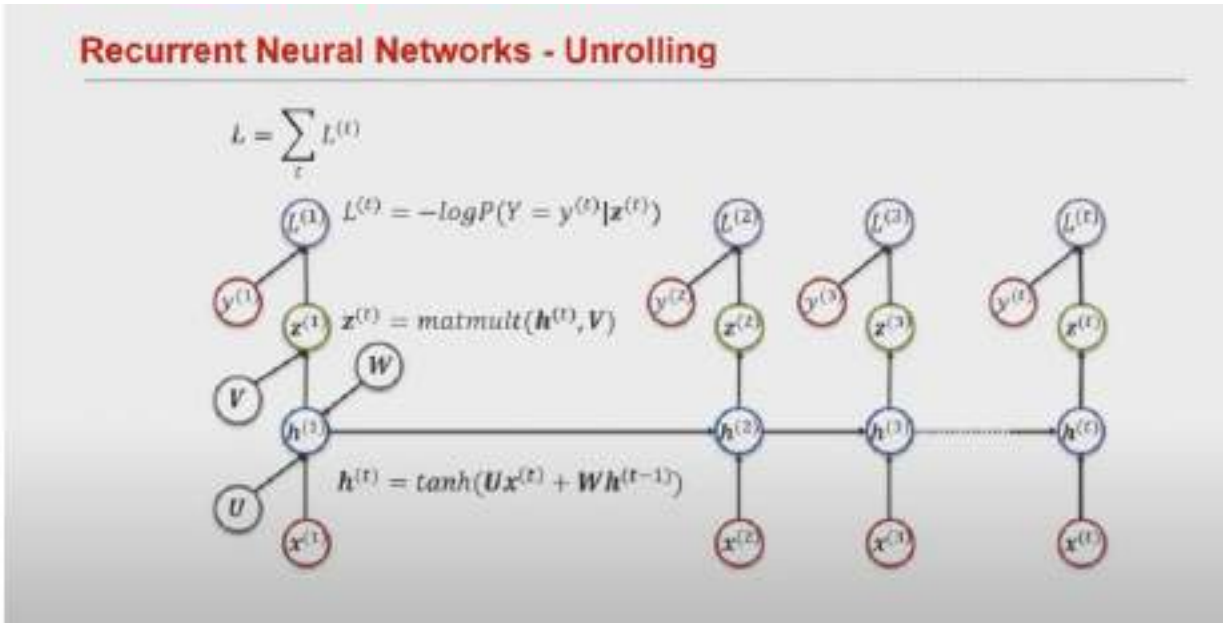


RNN for Sequence Prediction

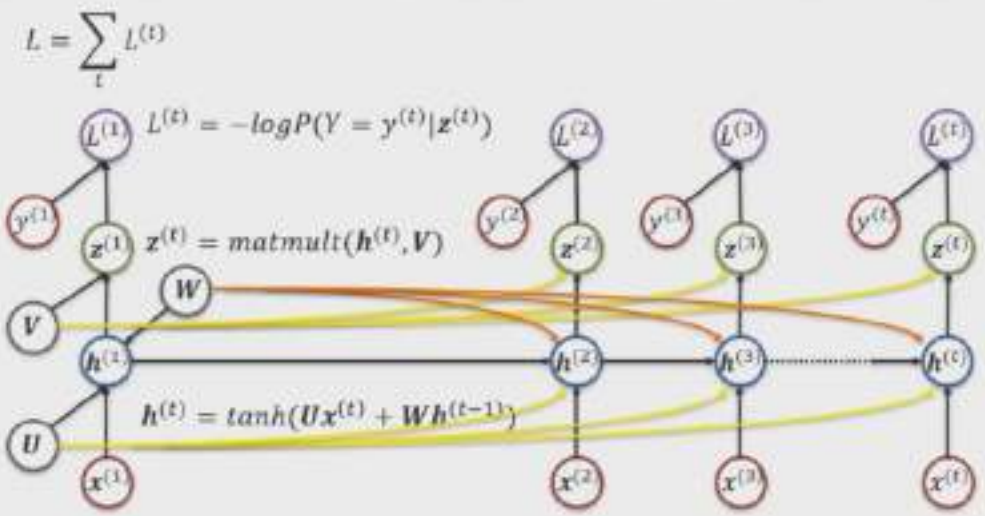


What is the loss? $L = \frac{1}{N}\sum_t L^{(t)} = \frac{1}{N}\sum_t -log P(Y = y^{(t)} | x^{(t)})$

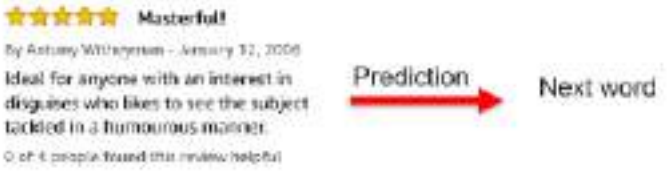
Recurrent Neural Networks - Unrolling



Recurrent Neural Networks - Unrolling



Sentence Modeling: Language Model



Language Model Application: Speech Recognition

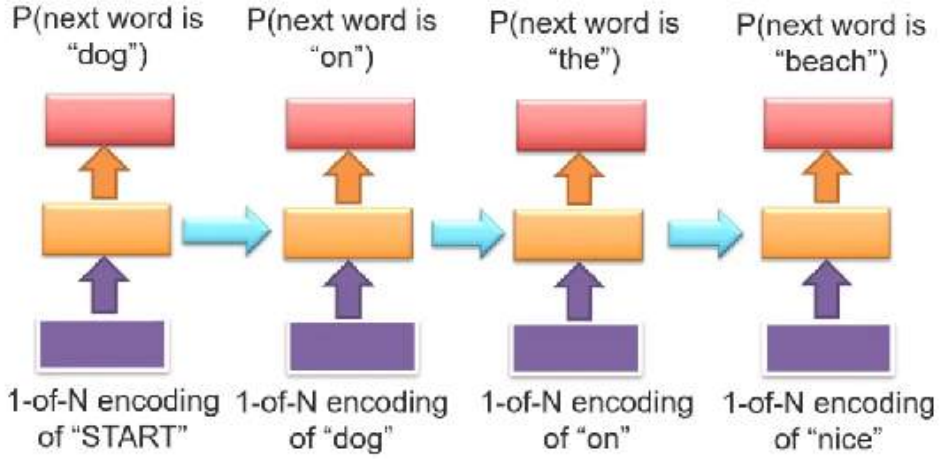
$$\arg\max_{wordsequence} P(wordsequence \mid acoustics) =$$

$$\arg\max_{wordsequence} \frac{P(acoustics \mid wordsequence) \times P(wordsequence)}{P(acoustics)}$$

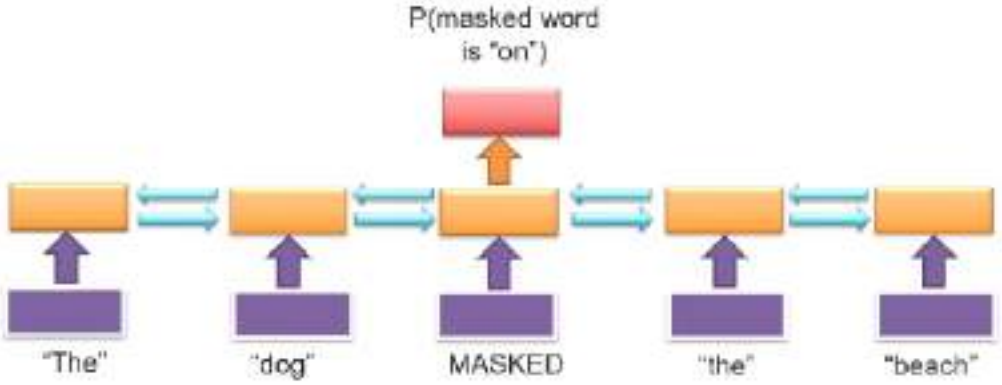
$$\arg\max_{wordsequence} P(acoustics \mid wordsequence) \times P(wordsequence)$$

Language model

RNN for Language Model

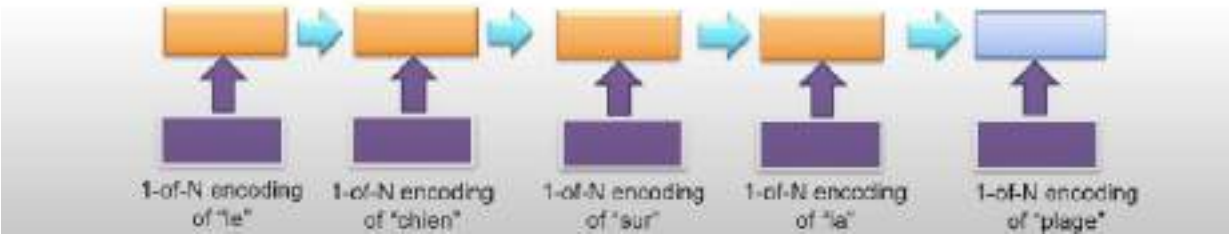


Pre-training and "Masking"

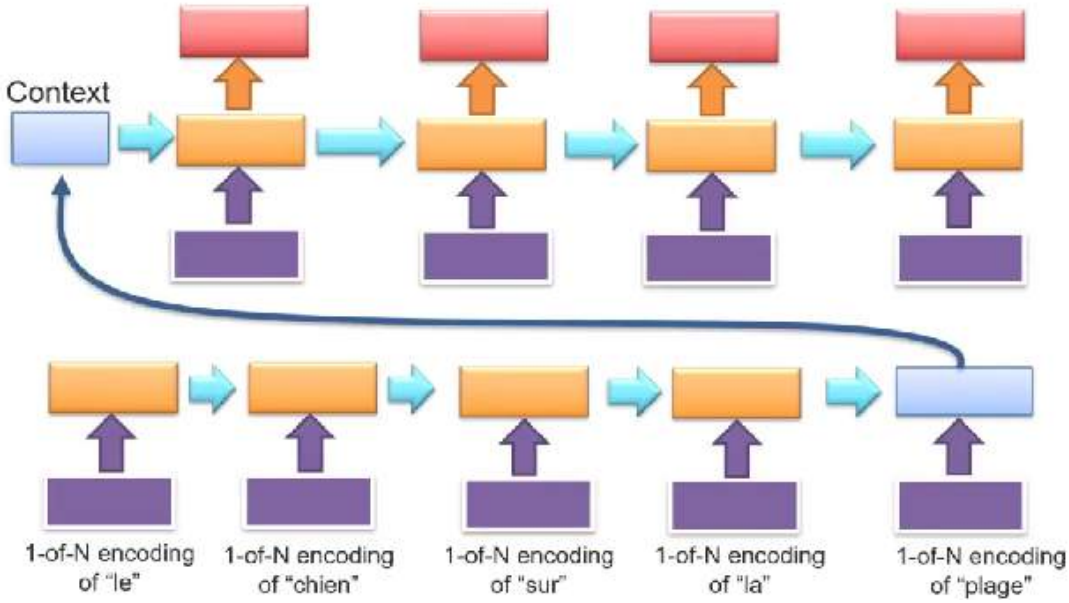


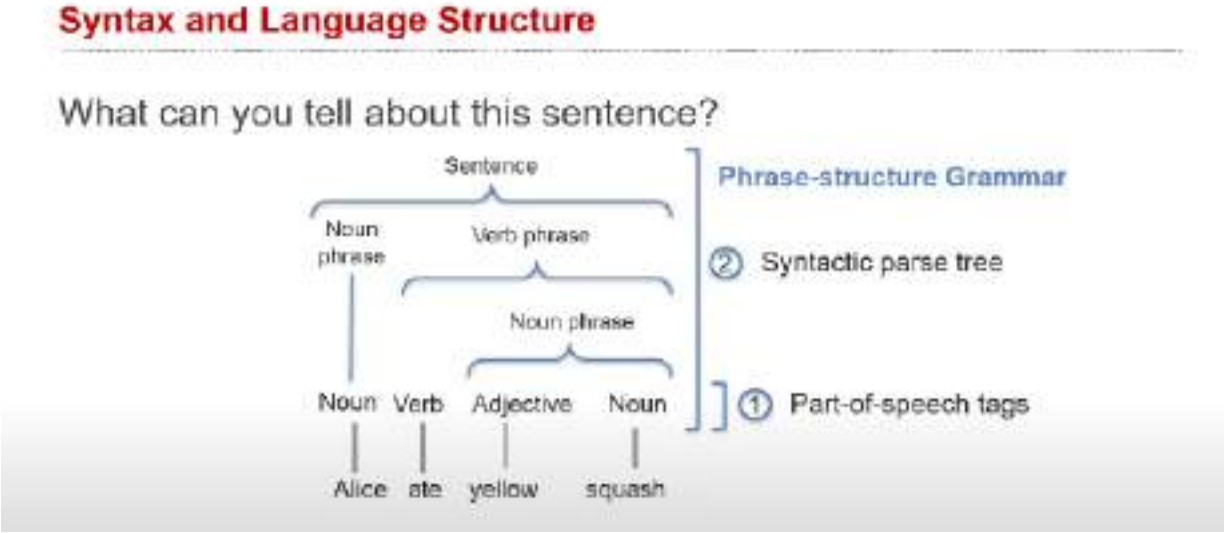
RNN-based for Machine Translation

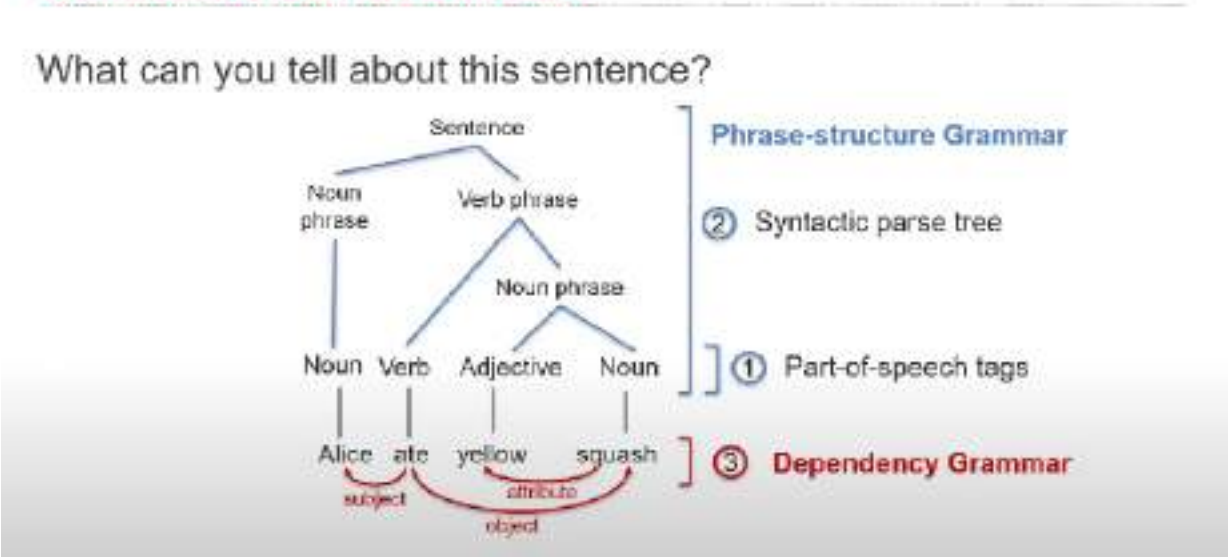
Le chien sur la plage → The dog on the beach



Encoder-Decoder Architecture

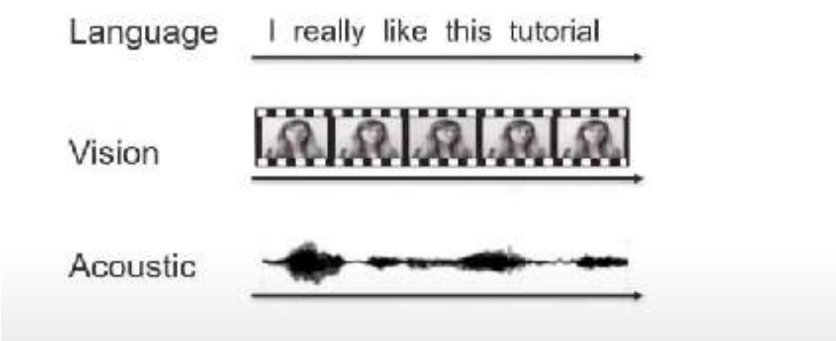






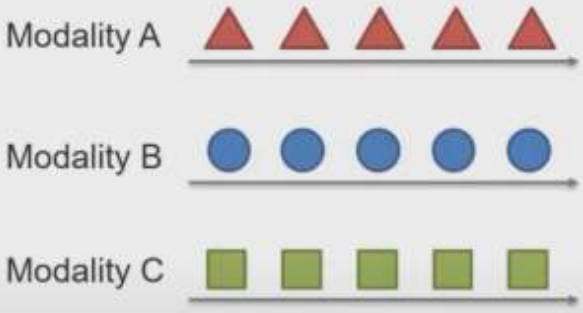
Lecture 3.2:

Multimodal Representations

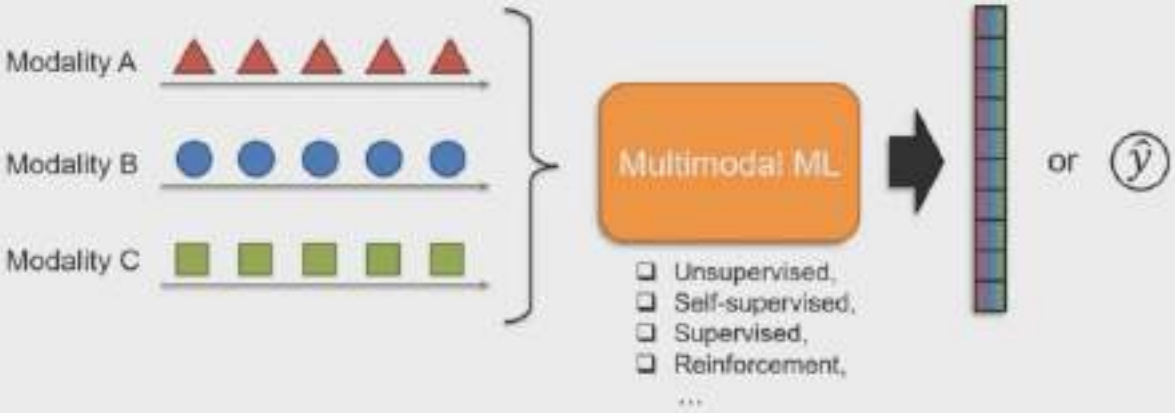


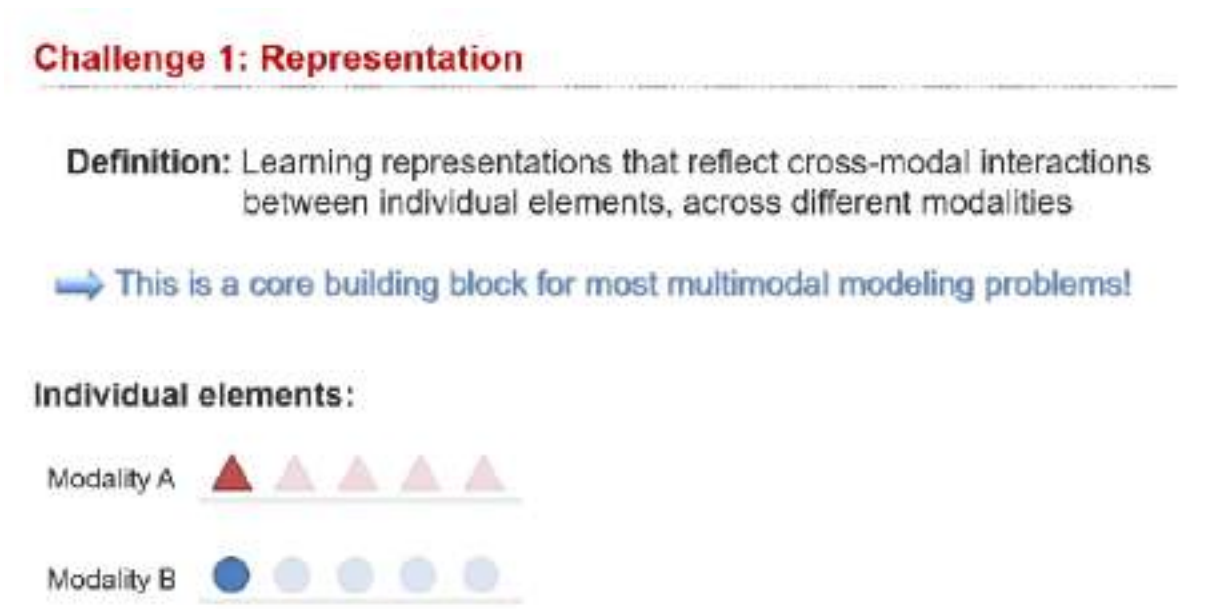
This could be written as :

Multimodal Machine Learning

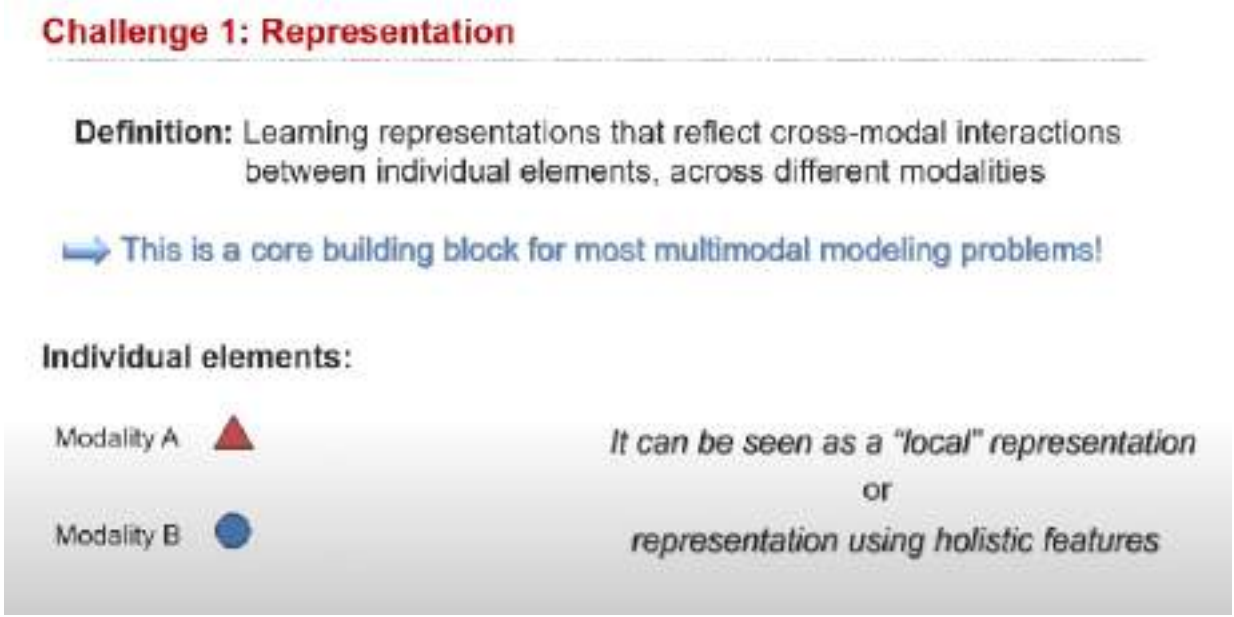


Multimodal Machine Learning



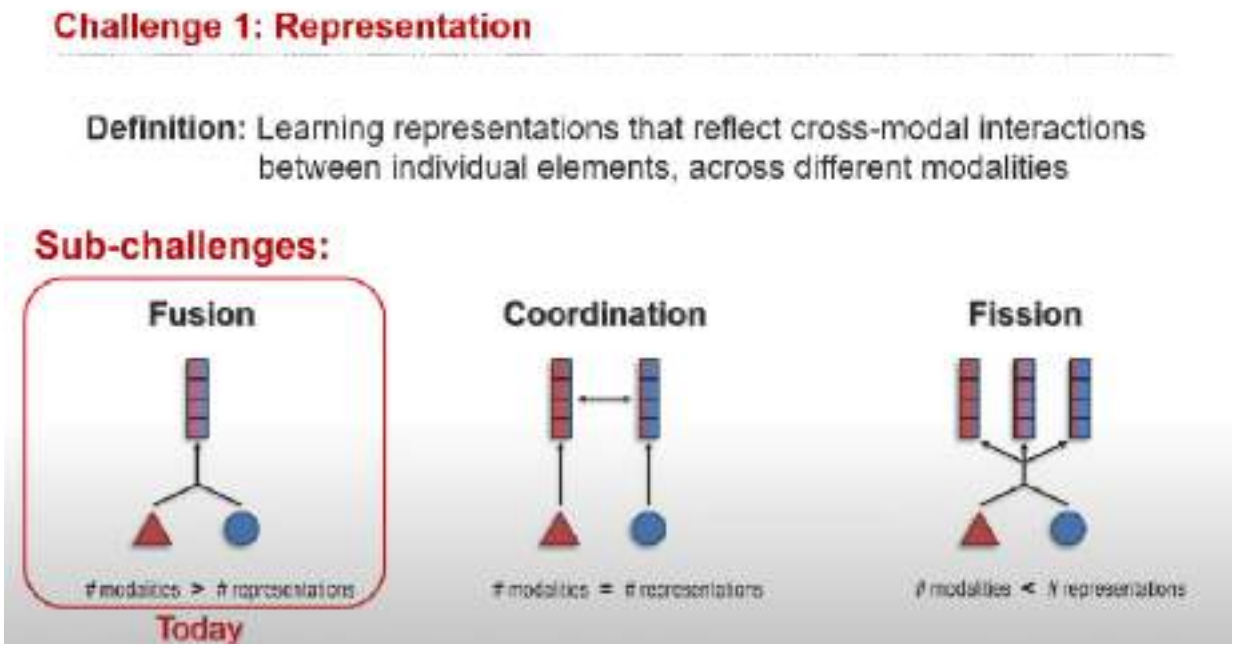


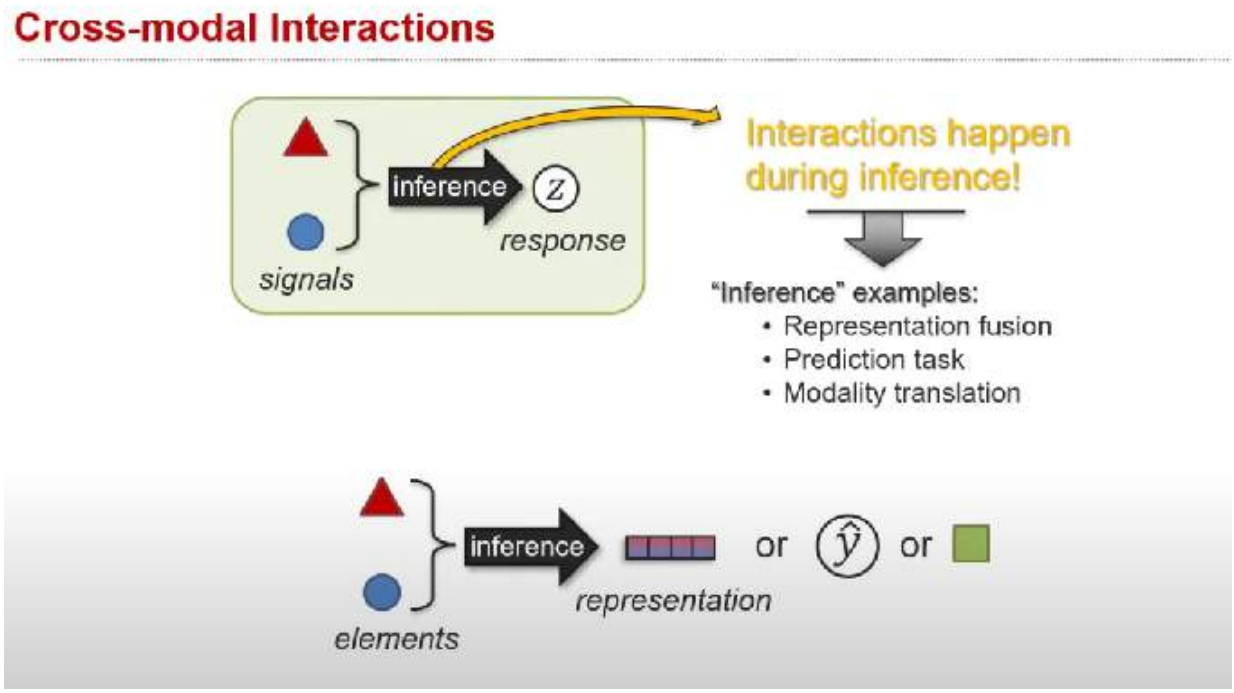
An individual element is a word for a sentence and a image for A



Fusion :In this case the number of input modalities is greater than output representations. In this case we are trying to condense the information together or fuse

the information together
Coordination:In this case we are exchanging information between modalities
Fission:





Interconnected Modalities

Types of interaction responses?
(a taxonomy)

Unimodal Non-redundancy

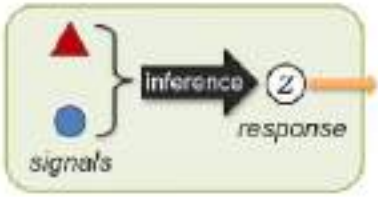
Is this a living room?

Inference → *Yes!*

A teacup on the right of a laptop in a clean room.

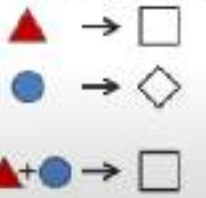
Inference → *No, probably study room.*

Interconnected Modalities



Types of interaction responses?
(a taxonomy)

Unimodal Non-redundancy



Is this a living room?

A teacup on the right of a laptop in a clean room.

Inference → ***Yes!***

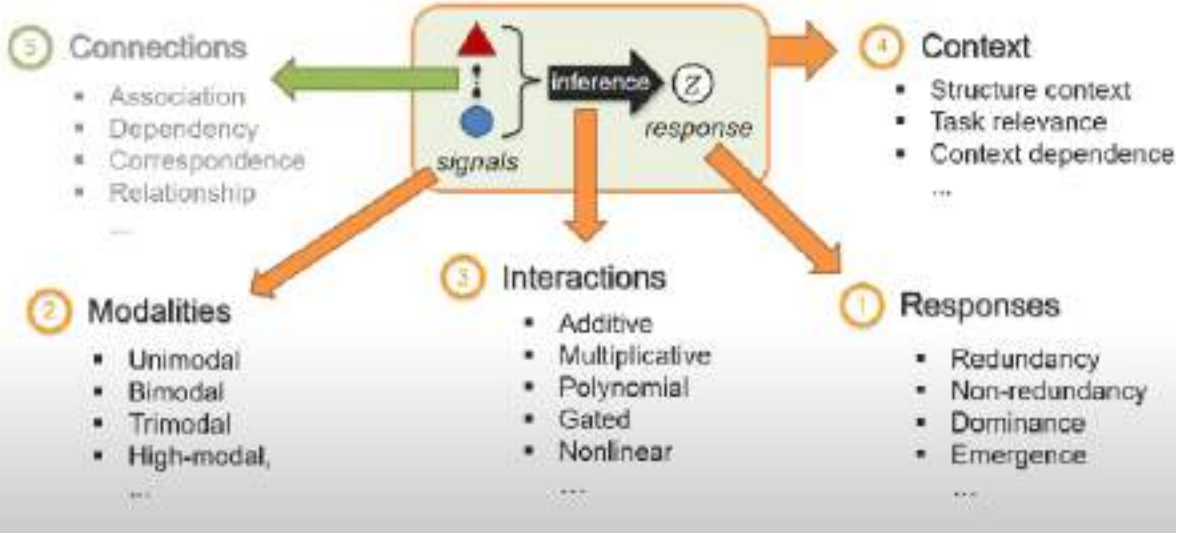
Multimodal dominance

Taxonomy of Interaction Responses – A Behavioral Science View

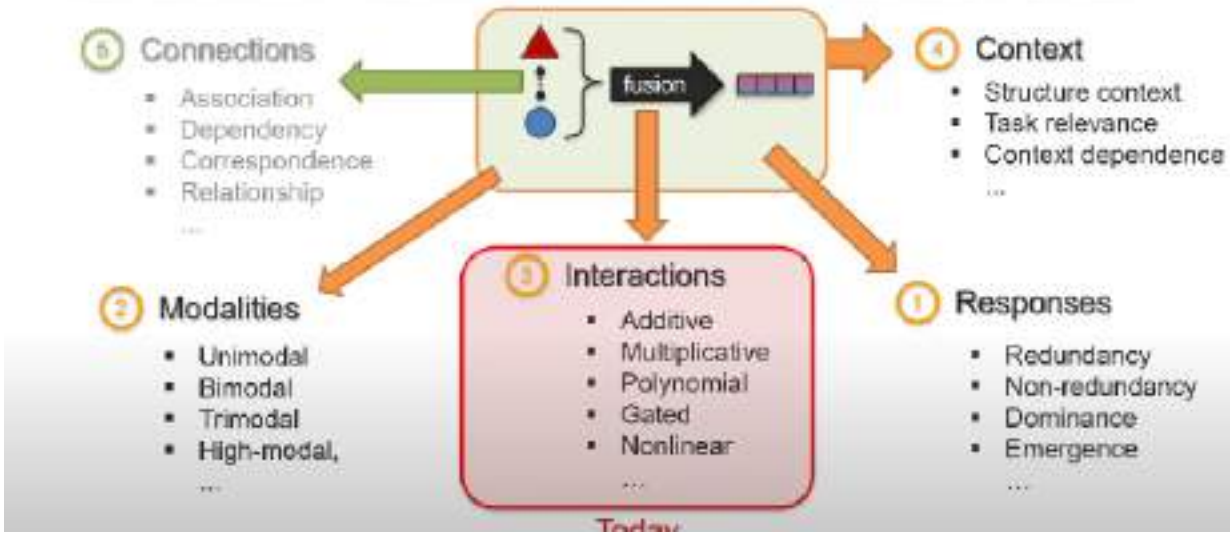
Multimodal Communication

	signal	response	signal	response	
Redundancy	a → □		a+b → □		Equivalence
	b → □		a+b → ▭		Enhancement
Nonredundancy	a → □		a+b → □ and ○		Independence
	b → ○		a+b → □		Dominance
			a+b → ▭ (or □)		Modulation
			a+b → △		Emergence

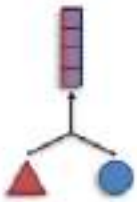
Cross-modal Interactions – A Taxonomy



Cross-modal Interactions – Representation Fusion



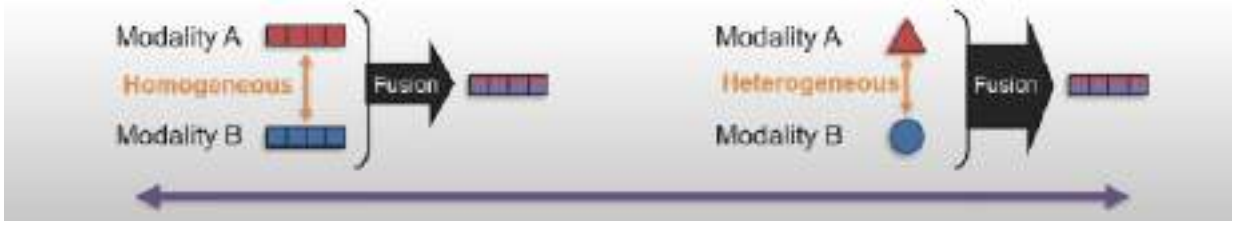
Sub-Challenge 1a: Representation Fusion



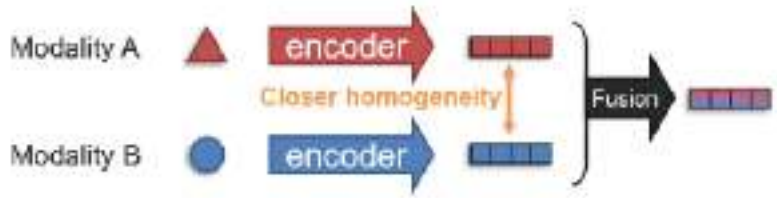
Definition: Learn a joint representation that models cross-modal interactions between individual elements of different modalities

Basic fusion:

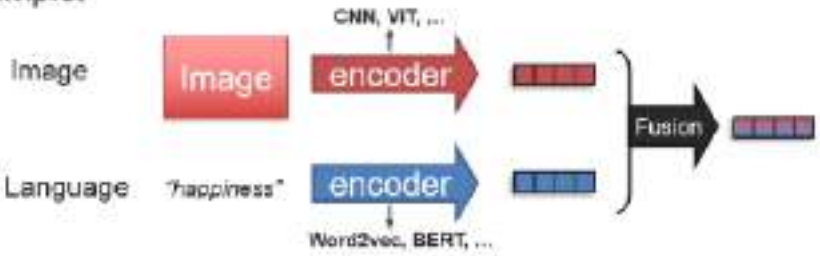
Raw-modality fusion:



Fusion with Unimodal Encoders

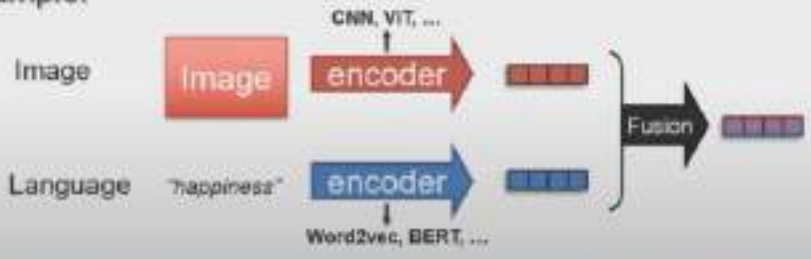


Example:



Fusion with Unimodal Encoders

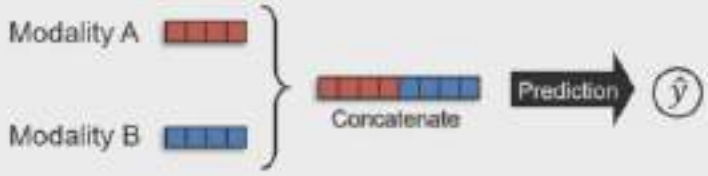
Example:



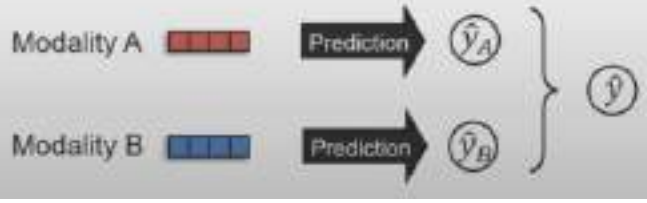
➥ Unimodal encoders can be jointly learned with fusion network, or pre-trained

Early and Late Fusion – A historical View

Early fusion:

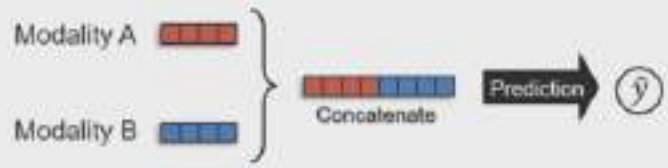


Late fusion:

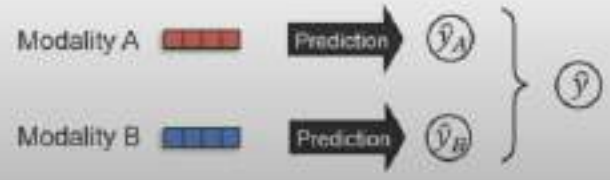


Early and Late Fusion – A historical View

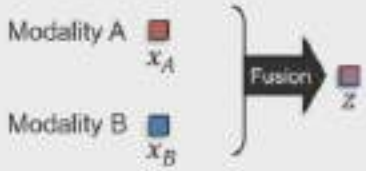
Early fusion:



Late fusion:



Basic Concepts for Representation Fusion (aka, Basic Fusion)



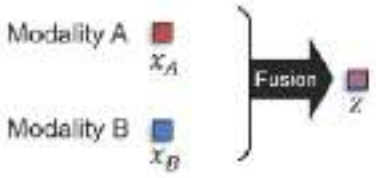
Goal: Model *cross-modal interactions* between the multimodal elements

➥ Let's study the univariate case first
(only 1-dimensional features)

Linear regression:

$$z = w_0 + w_1 x_A + w_2 x_B + w_3 (x_A \times x_b) + \epsilon$$

Basic Concepts for Representation Fusion (aka, Basic Fusion)



Goal: Model *cross-modal interactions* between the multimodal elements

➥ Let's study the univariate case first
(only 1-dimensional features)

Linear regression:

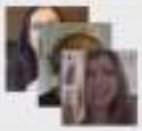
$$z = w_0 + \underbrace{w_1 x_A + w_2 x_B}_{\text{Additive terms}} + \underbrace{w_3 (x_A \times x_b)}_{\text{Multiplicative term}} + \underbrace{\epsilon}_{\text{error (residual term)}}$$

w_0: intercept (bias term)

Linear Regression

Linear regression is used to test research hypotheses, over a whole dataset

300 book reviews



y: audience score

x_A: percentage of smiling

x_B: professional status
(0=non-critic, 1=critic)

H1: *Does smiling reveal what the audience score was?*

H2: *Does the effect of smiling depend on professional status?*

Linear regression:

$$y = w_0 + \underbrace{w_1 x_A + w_2 x_B}_{\text{Additive terms}} + \underbrace{w_3 (x_A \times x_b)}_{\text{Multiplicative term}} + \underbrace{\epsilon}_{\text{error (residual term)}}$$

w_0: intercept (bias term)

w_0: average score when x_A and x_B are zero

w_1: effect from x_A variable only

w_2: effect from x_B variable only

w_3: effect from x_A and x_B interaction only

ϵ: residual not modeled by w_0, w_1, w_2 or w_3

Linear Regression

Linear regression is used to test research hypotheses, over a whole dataset

300 book reviews

y: audience score

x_A: percentage of smiling

x_B: professional status
(0=non-critic, 1=critic)

H1: Does smiling reveal what the audience score was?

H2: Does the effect of smiling depend on professional status?

Linear regression:

$$z = w_0 + \boxed{w_1} x_A + \epsilon$$

slope

Confidence interval: *"95% confident that w parameter is contained within this interval"*

	Estimate	95% CI
w_0	4.63	[4.20, 5.06]
w_1	1.20	[0.83, 1.57]

Confidence interval does not contain 0, so effect is significant

p-values would be another way to test hypothesis

Linear Regression

Linear regression is used to test research hypotheses, over a whole dataset

300 book reviews

y: audience score

x_A: percentage of smiling

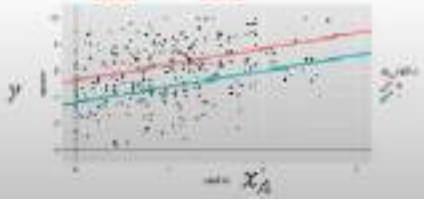
x_B: professional status
(0=non-critic, 1=critic)

H1: Does smiling reveal what the audience score was?

H2: Does the effect of smiling depend on professional status?

Linear regression:

$$z = w_0 + \boxed{w_1} x_A + \boxed{w_2} x_B + \epsilon$$



	Estimate	95% CI	
w_0	5.29	[4.86, 5.73]	
w_1	1.19	[0.85, 1.53]	Positive effect
w_2	−1.69	[−2.14, −1.24]	Negative effect

Linear Regression

Linear regression is used to test research hypotheses, over a whole dataset

300 book reviews

y: audience score

x_A: percentage of smiling

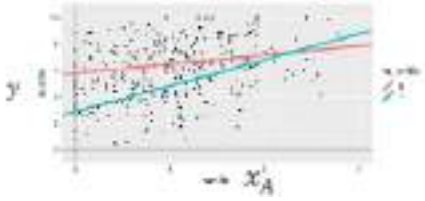
x_B: professional status
(0=non-critic, 1=critic)

H1: Does smiling reveal what the audience score was?

***H2:** Does the effect of smiling depend on professional status?*

Linear regression:

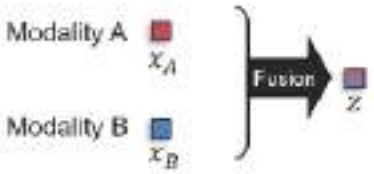
$$z = w_0 + w_1 x_A + w_2 x_B + w_3 (x_A \times x_b) + \epsilon$$



	Estimate	95% CI
w_0	5.79	[5.29, 6.29]
w_1	0.68	[0.25, 1.11]
w_2	−2.94	[−3.73, −2.15]
w_3	1.29	[0.61, 1.97]

➡ Multiplicative interaction!

Basic Concepts for Representation Fusion (aka, Basic Fusion)



Goal: Model *cross-modal interactions* between the multimodal elements

➡ Let's study the univariate case first
(only 1-dimensional features)

Linear regression:

$$z = w_0 + w_1 x_A + w_2 x_B + w_3 (x_A \times x_b) + \epsilon$$

intercept (bias term) — Additive terms — Multiplicative term — error (residual term)

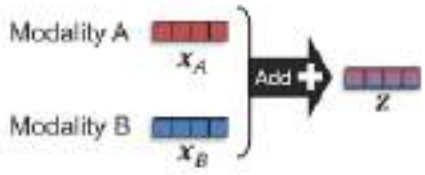
① Additive terms:
$$z = w_1 x_A + w_2 x_B + \epsilon$$

② Multiplicative "interaction" term:
$$z = w_3 (x_A \times x_b) + \epsilon$$

③ Additive and multiplicative terms:
$$z = w_1 x_A + w_2 x_B + w_3 (x_A \times x_b) + \epsilon$$

Additive Fusion → Back to multivariate case!

(multi-dimensional features)

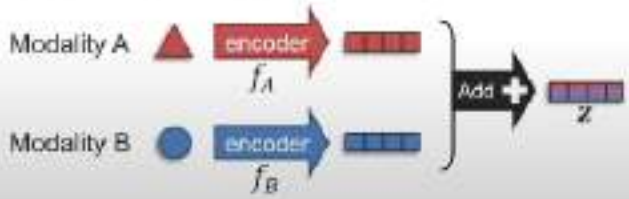


Additive fusion:

$$z = w_1 x_A + w_2 x_B$$

→ 1-layer neural network can be seen as additive

With unimodal encoders:



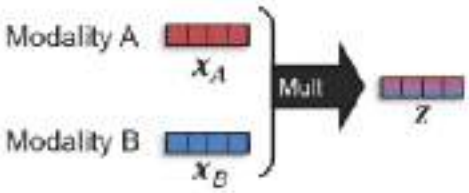
Additive fusion:

$$z = f_A(\blacktriangle) + f_B(\bullet)$$

→ It could be seen as an ensemble approach

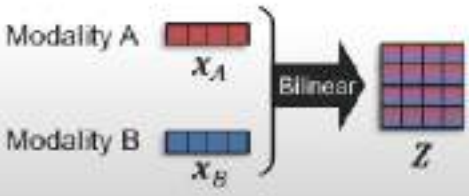
(late fusion)

Multiplicative Fusion



Simple multiplicative fusion:

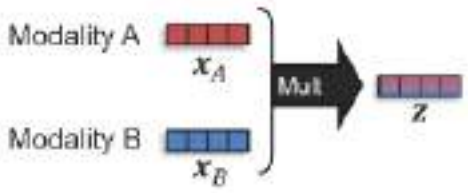
$$z = w(x_A \times x_B)$$



Bilinear Fusion:

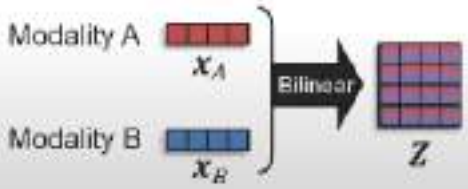
$$Z = W(x_A^T \cdot x_B)$$

Multiplicative Fusion



Simple multiplicative fusion:

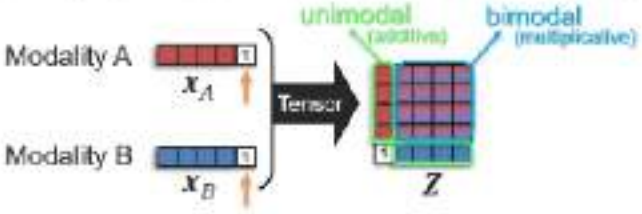
$$z = w(x_A \times x_B)$$



Bilinear Fusion:

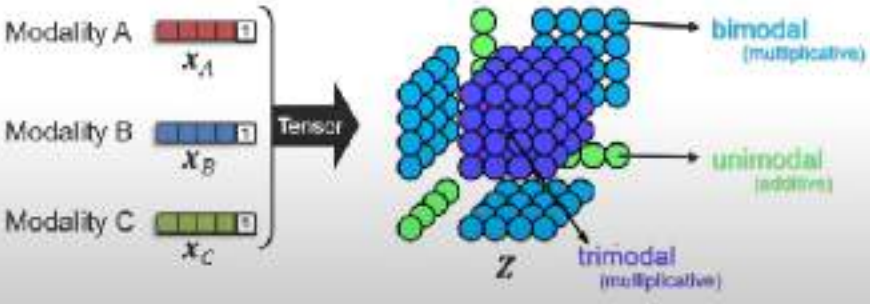
$$Z = W(x_A^T \cdot x_B)$$

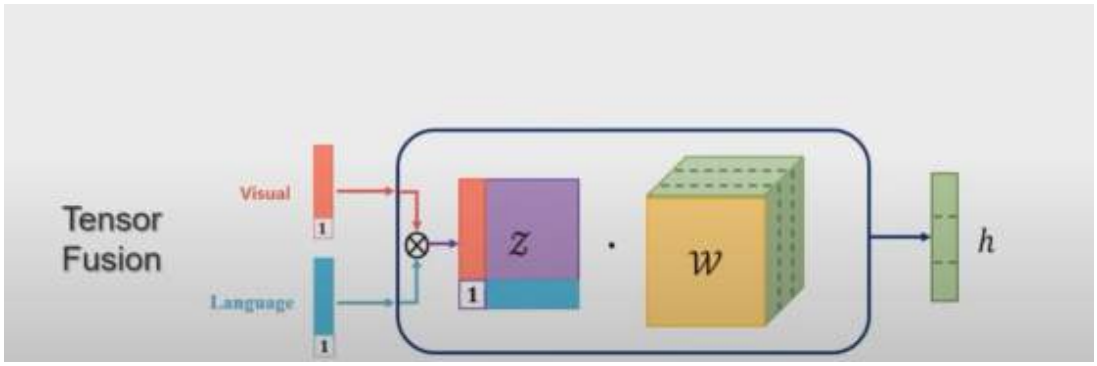
Tensor Fusion

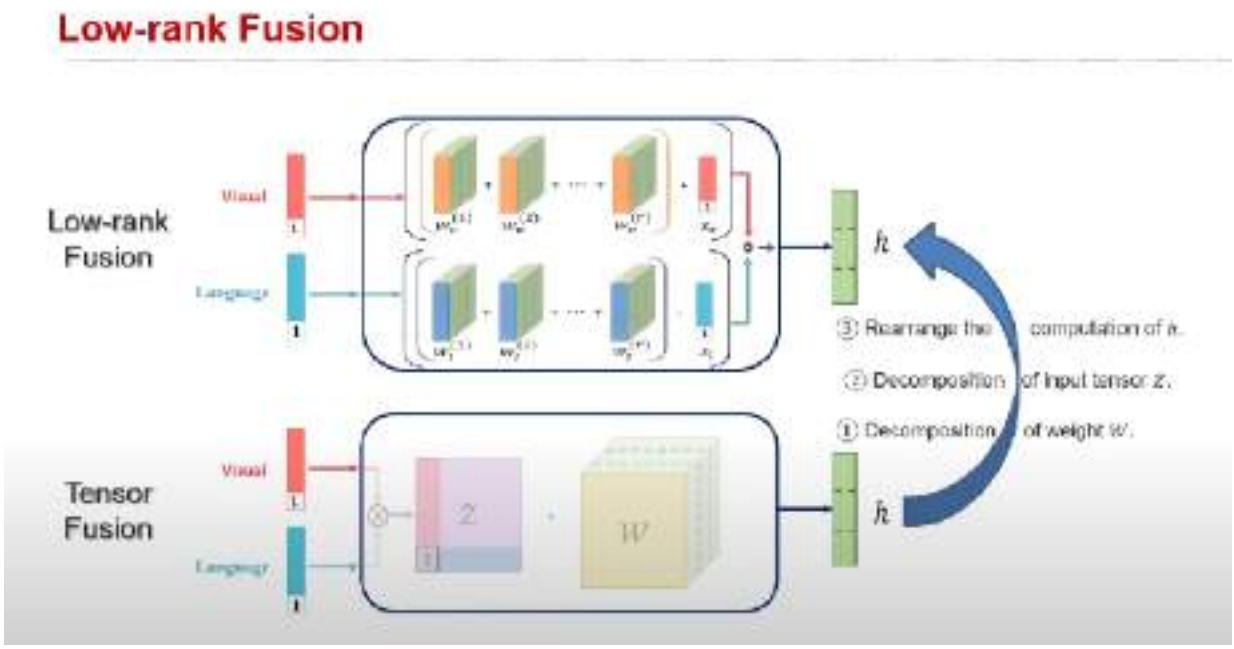


Tensor Fusion (bimodal):

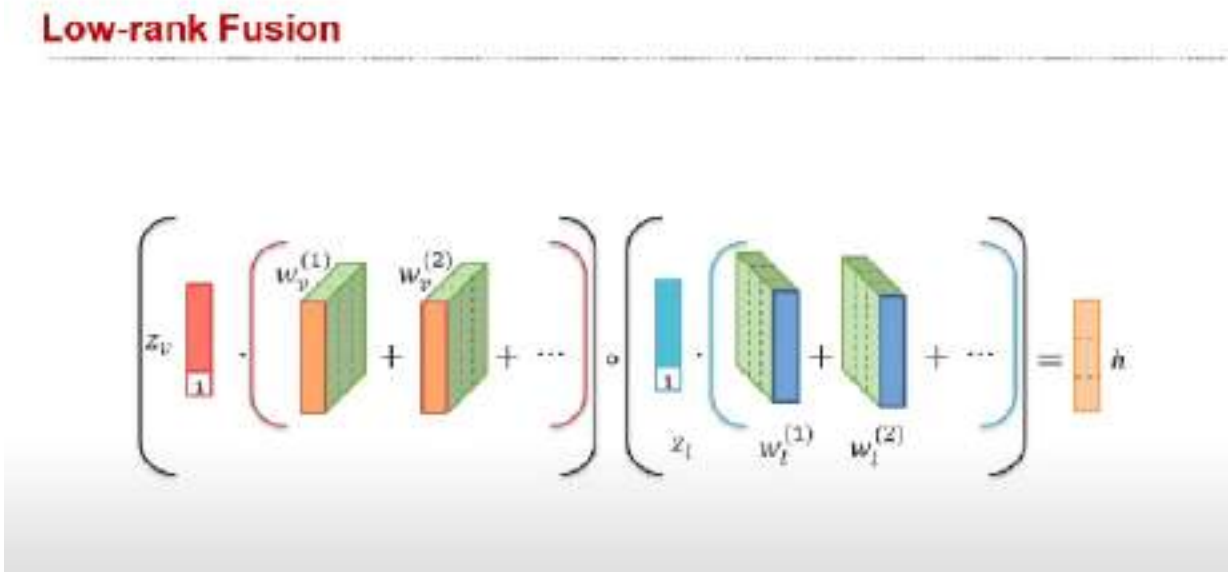
$$Z = w([x_A \quad 1]^T \cdot [x_B \quad 1])$$



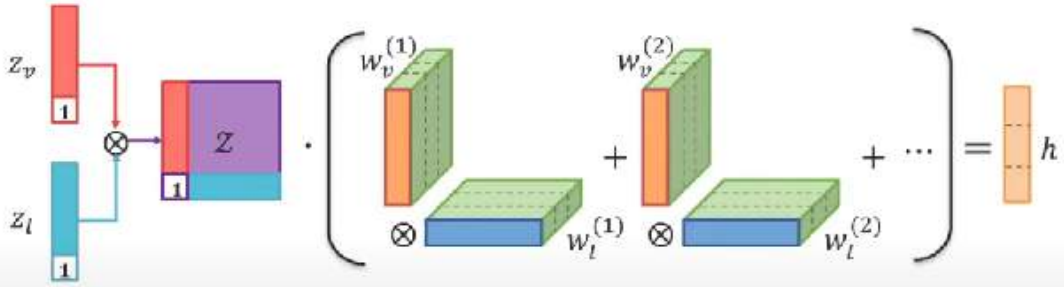




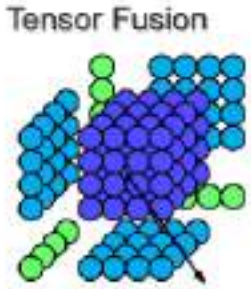
We decompose the W matrix into the following:



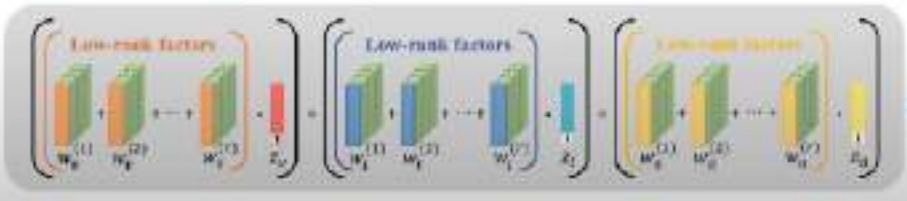
Low-rank Fusion



Low-rank Fusion with Trimodal Input



Low-rank Fusion :



Going Beyond Additive and Multiplicative Fusion

Additive interaction:

$z = w_1 x_A + w_2 x_B$ ⟸ First-order polynomial

Additive and multiplicative interaction:

$z = w_1 x_A + w_2 x_B + w_3 (x_A \times x_B)$ ⟸ Second-order polynomial

Trimodal fusion (e.g., tensor fusion):

$$z = \underbrace{w_1 x_A + w_2 x_B + w_3 x_C}_{\text{Unimodal terms (first-order)}} + \underbrace{w_4 (x_A \times x_C) + w_5 (x_A \times x_C) + w_6 (x_B \times x_C)}_{\text{Bimodal terms (second-order)}} + \underbrace{w_7 (x_A \times x_B \times x_C)}_{\text{Trimodal terms (third-order)}}$$

Can we add higher-order interaction terms?

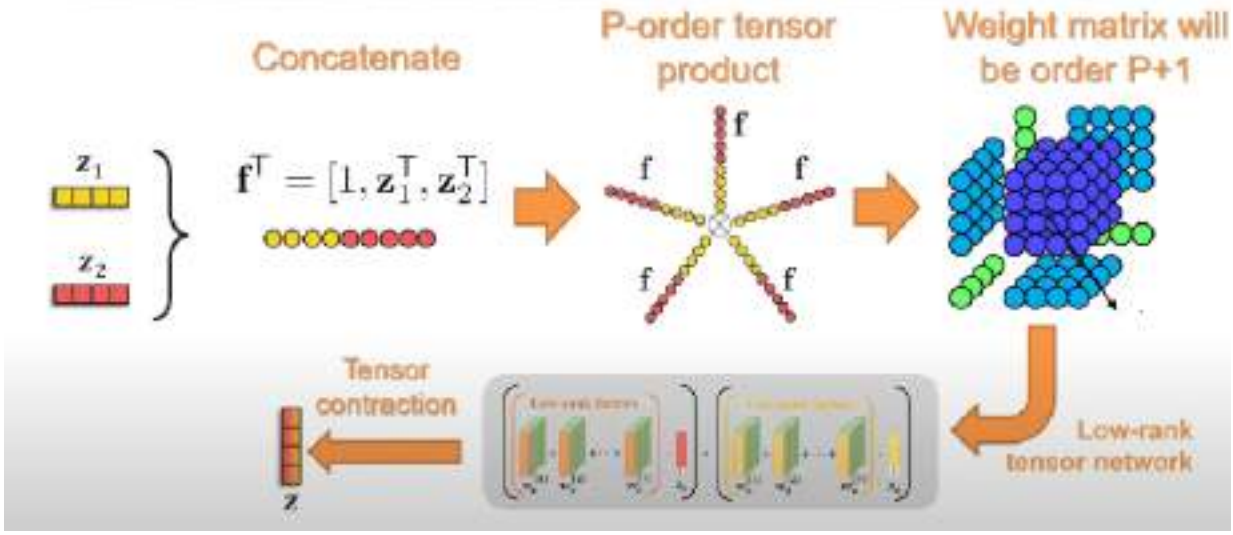
For example:

$+ w_8 (x_A^2 \times x_B^3 \times x_C^0)$

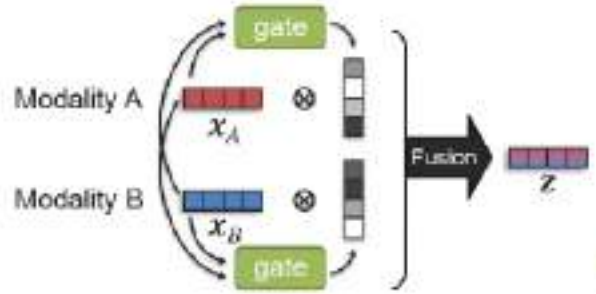
$+ w_9 (x_A^2 \times x_B)$

$+ w_{10} (x_B^3 \times x_C^2)$

High-Order Polynomial Fusion



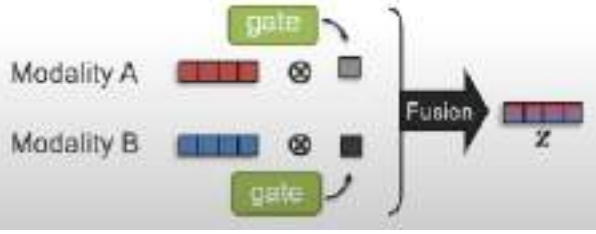
Gated Fusion



Example with additive fusion:

$$z = g_A(x_A, x_B) \cdot x_A + g_B(x_A, x_B) \cdot x_B$$

→ g_A and g_B can be seen as attention functions



→ Gating output can be one weight for the whole modality

Gating Module (aka, attention module)

"Neural network designed to mask unwanted signal from propagating forward" (gating)

…or with a more positive view:

"Neural network designed to select preferable signal to move forward" (attention)

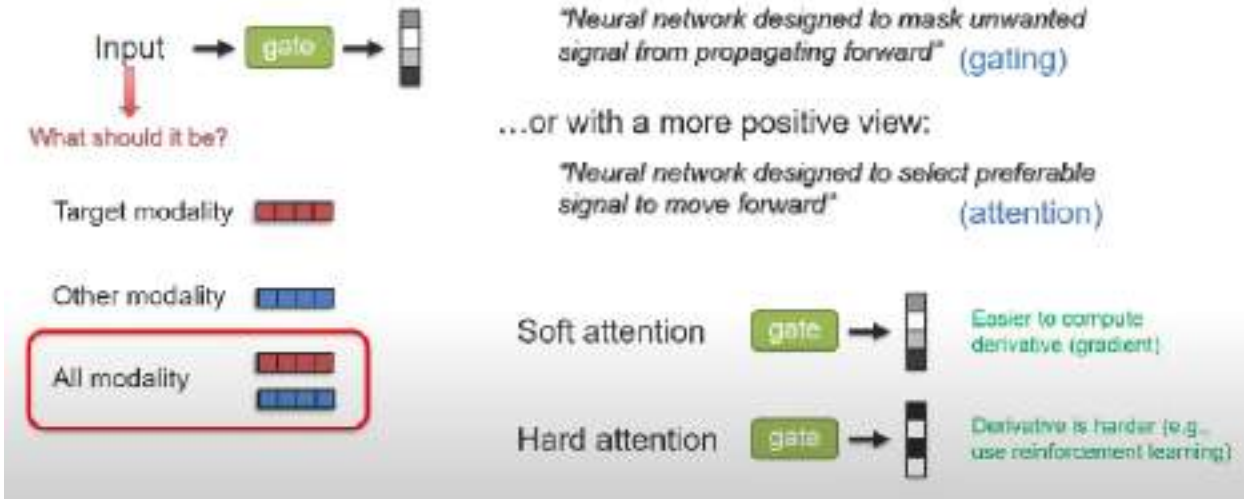
Gating Module (aka, attention module)

"Neural network designed to mask unwanted signal from propagating forward" (gating)

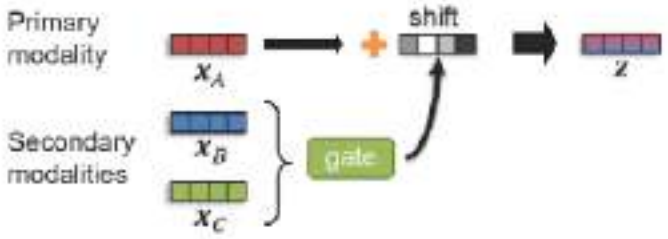
…or with a more positive view:

"Neural network designed to select preferable signal to move forward" (attention)

Gating Module (aka, attention module)



Modality-Shifting Fusion

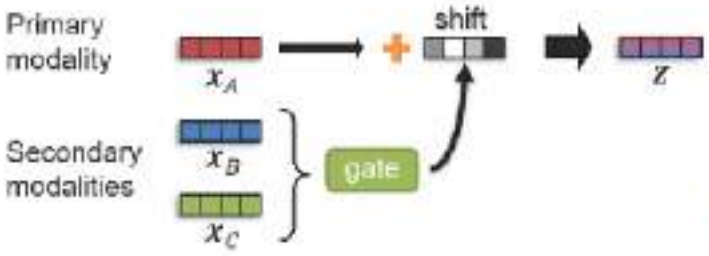


Example with language modality:

Primary modality: language

Secondary modalities: acoustic and visual

Modality-Shifting Fusion

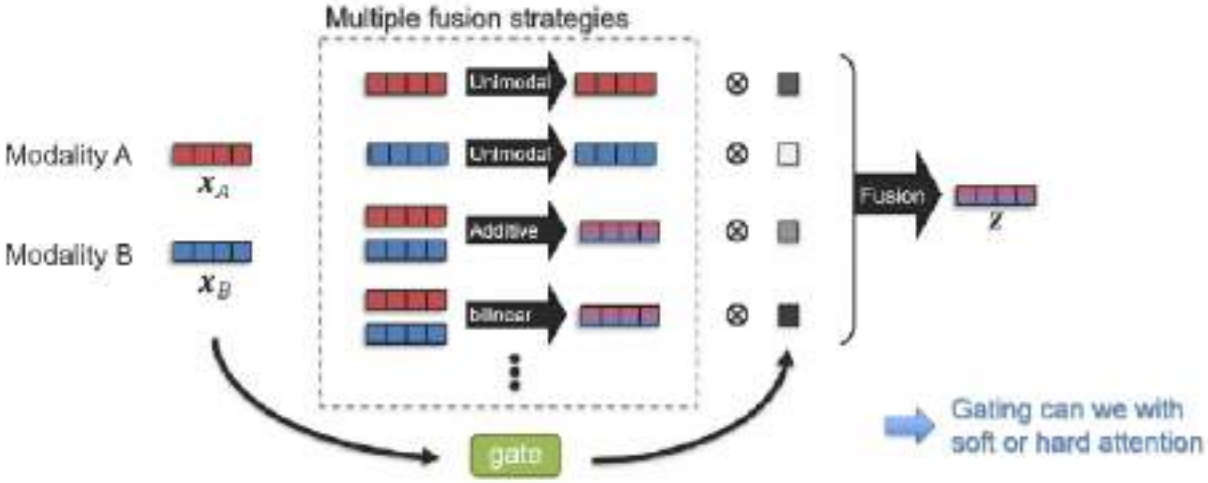


Example with language modality:

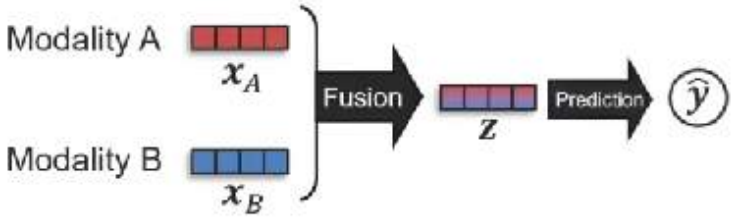
Primary modality: language

Secondary modalities: acoustic and visual

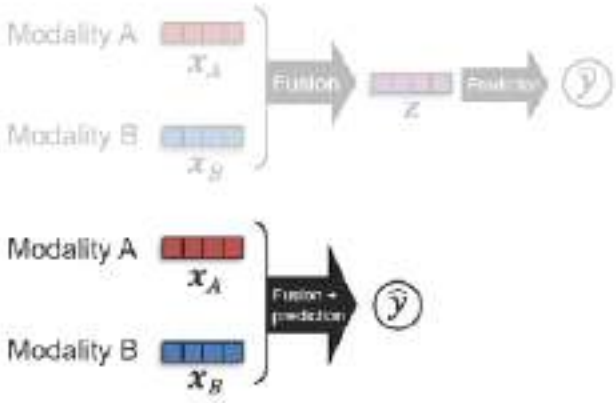
Dynamic Fusion



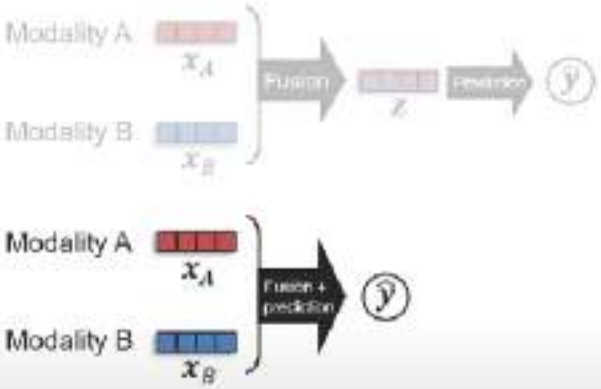
Nonlinear Fusion



Nonlinear Fusion



Nonlinear Fusion



Nonlinear fusion:

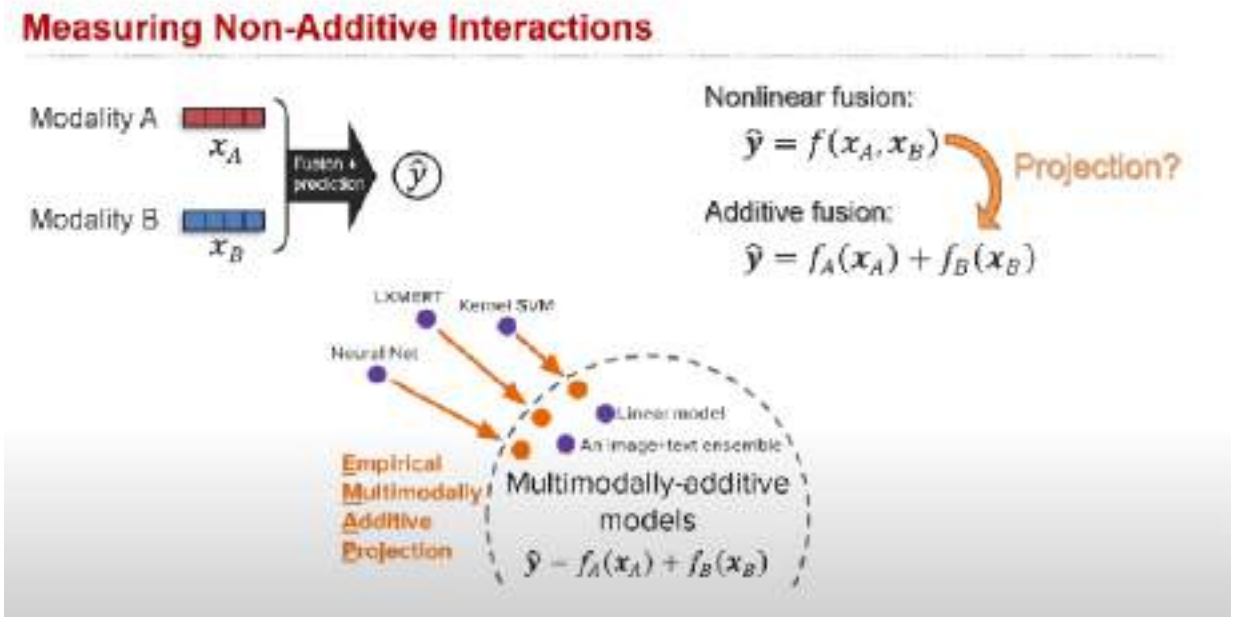
$$\hat{y} = f(x_A, x_B) \in \mathbb{R}^d$$

where f could be a multi-layer perceptron or any nonlinear model

➡ This could be seen as *early fusion*:

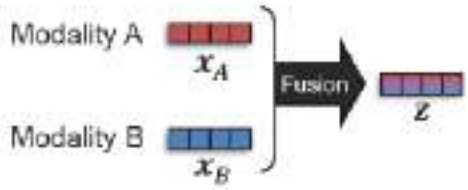
$$\hat{y} = f([x_A, x_B])$$

... but will our neural network learn the nonlinear interactions?


Measuring Non-Additive Interactions
Modality A
x_A
Modality B
x_B
Fusion + prediction
$\hat{y}$
Nonlinear fusion:
$\hat{y} = f(x_A, x_B)$
Projection?
Additive fusion:
$\hat{y} = f_A(x_A) + f_B(x_B)$
LIGHT
Kernel SVM
Neural Net
Linear model
An image+text ensemble
Empirical
Multimodally
Additive
Projection
Multimodally-additive models
$\hat{y} = f_A(x_A) + f_B(x_B)$

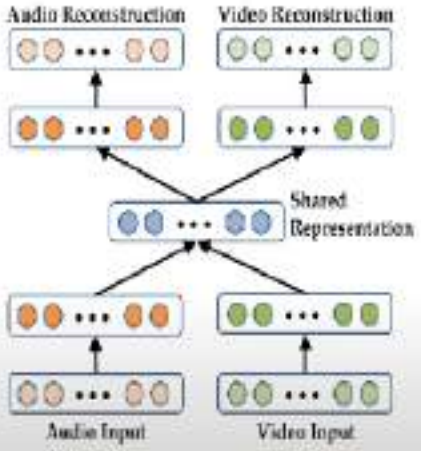
Lecture 4.1

Learning Fusion Representations

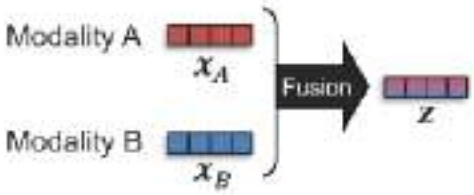


How to learn fusion models?

Multimodal Autoencoder

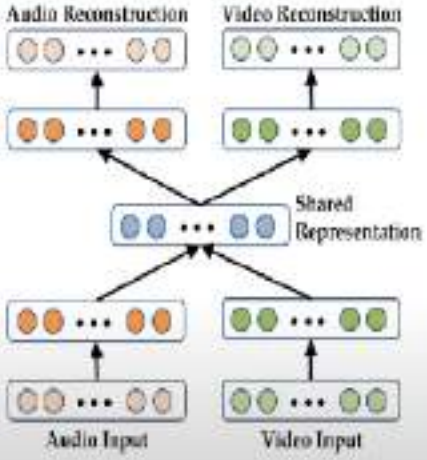


Learning Fusion Representations

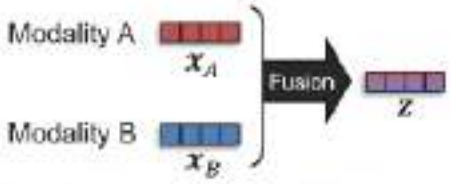


How to learn fusion models?

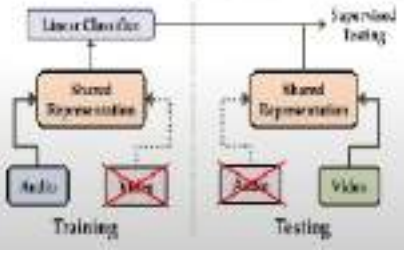
Multimodal Autoencoder



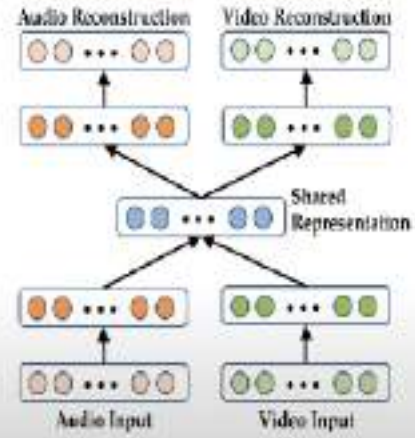
Learning Fusion Representations



Interesting experiment: "Hearing to see"
(zero-shot cross-modal adaptation)

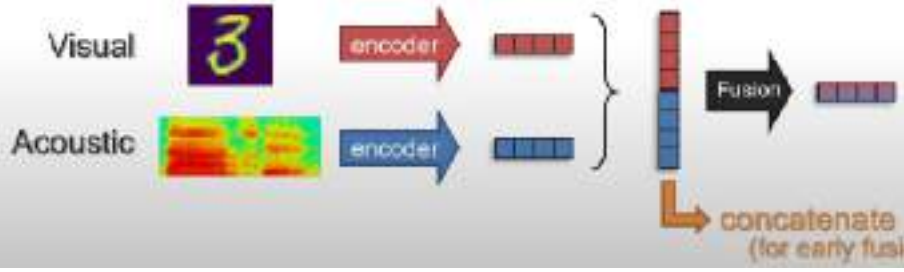


Multimodal Autoencoder



Fusion with Raw Modalities

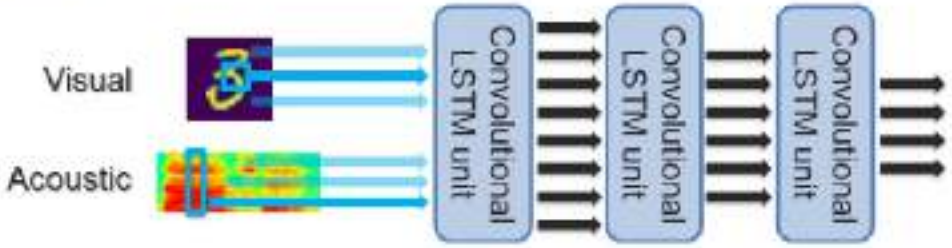
Example: From Early Fusion...



Fusion with Raw Modalities

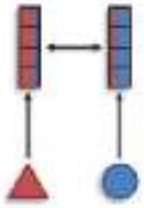
Open Challenge!

Example: From Early Fusion... to Very Early Fusion (inspired by human brain)



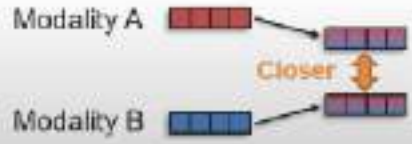
Representation Coordination :

Sub-Challenge 1b: Representation Coordination

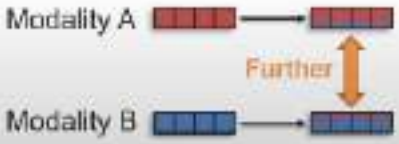


Definition: Learn multimodally-contextualized representations that are coordinated through their cross-modal interactions

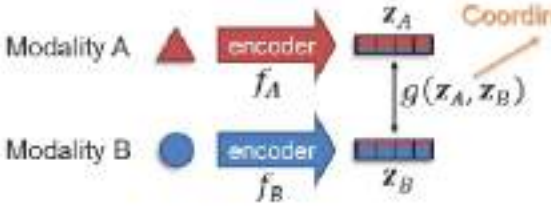
Strong Coordination:



Partial Coordination:



Coordination Function



Learning with coordination function:

$$\mathcal{L} = g(f_A(\blacktriangle), f_B(\bullet))$$

with model parameters θ_g, θ_{f_A} and θ_{f_B}

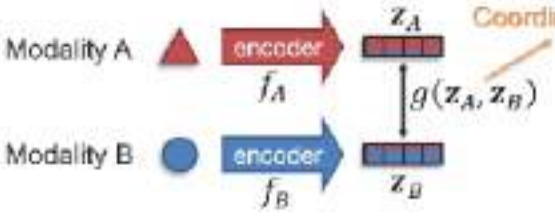
➡ Requires paired data

Examples of coordination function:

① Cosine similarity: $g(z_A, z_B) = \dfrac{z_A \cdot z_B}{\|z_A\|\|z_B\|}$ Strong coordination!

➡ For normalized inputs (e.g., $z_A - \bar{z_A}$), equivalent to *Pearson correlation coefficient*

Coordination Function



Learning with coordination function:

$$\mathcal{L} = g\big(f_A(\blacktriangle), f_B(\bullet)\big)$$

with model parameters θ_g, θ_{f_A} and θ_{f_B}

Examples of coordination function:

② Kernel similarity functions:

$$g(z_A, z_B) = k(z_A, z_B)$$

- Linear
- Polynomial
- Exponential
- RBF

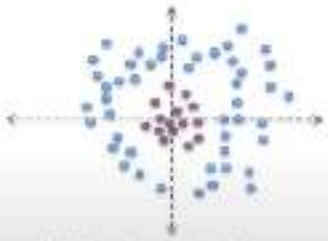
➡ All these examples bring relatively strong coordination between modalities

Kernel Function

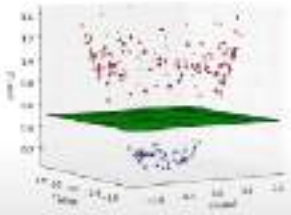
A kernel function: Acts as a similarity metric between data points

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j) = \langle \phi(x_i), \phi(x_j) \rangle$$

➡ $\phi(x)$ can be high-dimensional space!



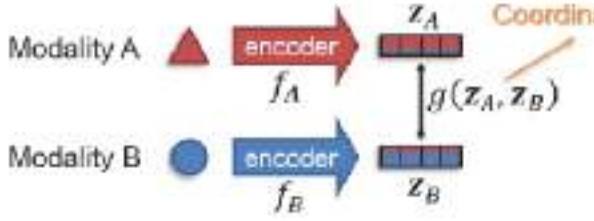
Not linearly separable in x space



Same data, but now linearly separable in $\phi(x)$ space

➡ Radial Basis Function (RBF) Kernel : $K(x_i, x_j) = \exp -\frac{1}{2\sigma^2}\|x_i - x_j\|^2$

Coordination Function



Learning with coordination function:

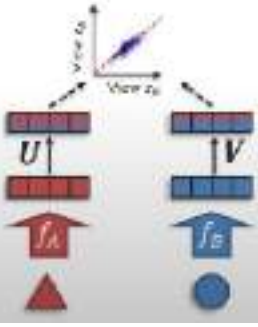
$$\mathcal{L} = g(f_A(\blacktriangle), f_B(\bullet))$$

with model parameters θ_g, θ_{f_A} and θ_{f_B}

Examples of coordination function:

③ Canonical Correlation Analysis (CCA):

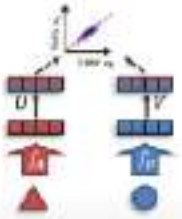
$$\underset{V,U,f_A,f_B}{\operatorname{argmax}}\, corr(z_A, z_B)$$



Correlated Projection

① Learn two linear projections, one for each view, that are maximally correlated:

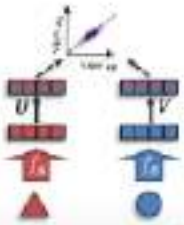
$$(u^*, v^*) = \underset{u,v}{\operatorname{argmax}}\, corr(u^T X, v^T Y)$$



Two views X, Y where same instances have the same color

⟹ Remember that X and Y consist of paired data

Canonical Correlation Analysis



(2) We want these multiple projection pairs to be orthogonal ("canonical") to each other:

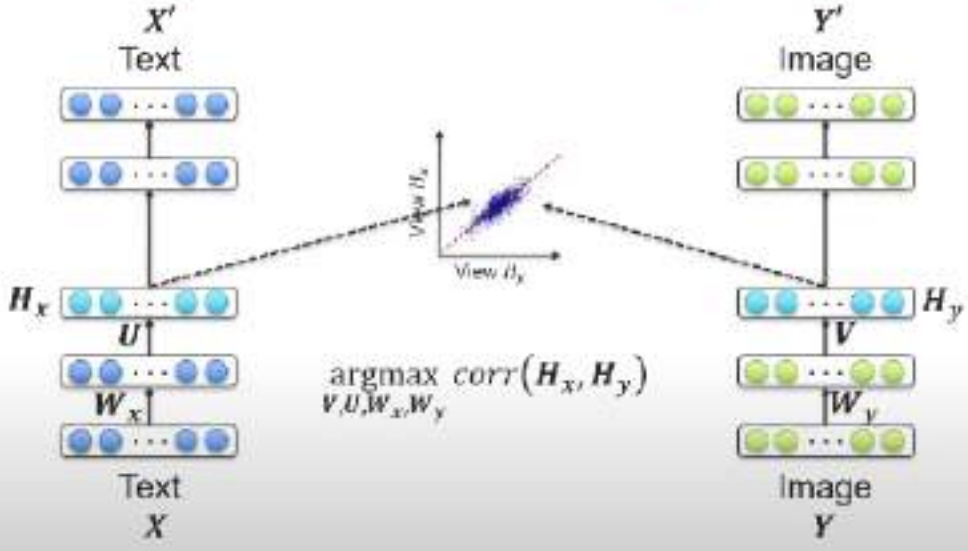
$$u_{(i)}^T \Sigma_{XY} v_{(j)} = u_{(j)}^T \Sigma_{XY} v_{(i)} = 0 \qquad \text{for } i \neq j$$

$$|U\Sigma_{XY}V| = tr(U\Sigma_{XY}V) \qquad \text{where } U = [u_{(1)}, u_{(2)}, \ldots, u_{(k)}]$$
$$\text{and } V = [v_{(1)}, v_{(2)}, \ldots, v_{(k)}]$$

(3) Since this objective function is invariant to scaling, we can constraint the projections to have unit variance:

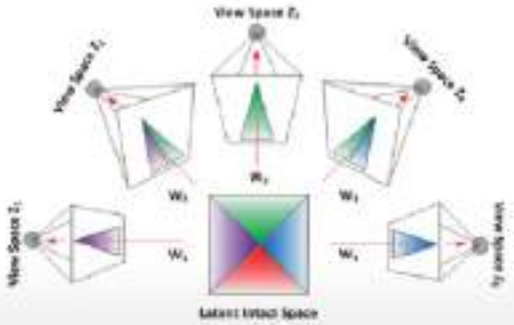
$$U^T \Sigma_{XX} U = I \qquad V^T \Sigma_{YY} V = I$$

Deep Canonically Correlated Autoencoders (DCCAE)



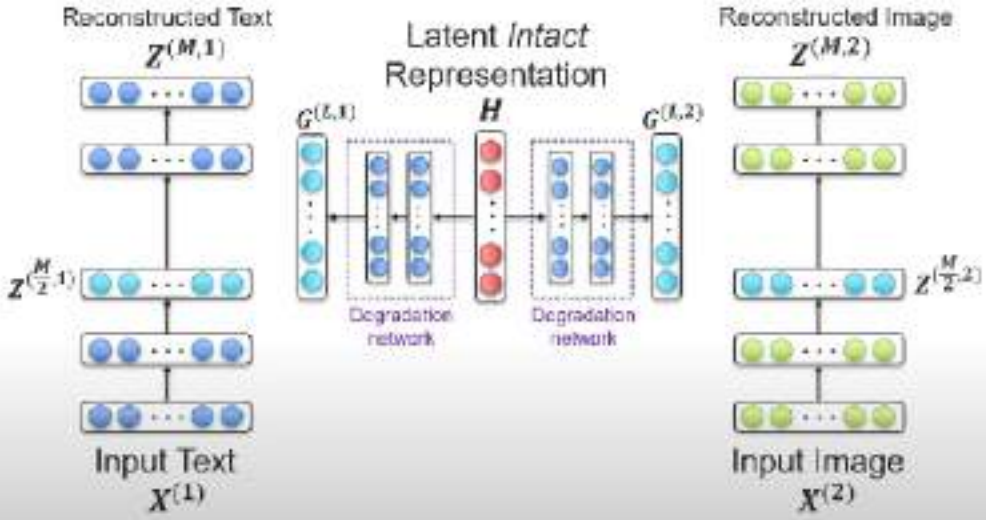
Multi-view Latent "Intact" Space

Given multiple views z_i from the same "object":

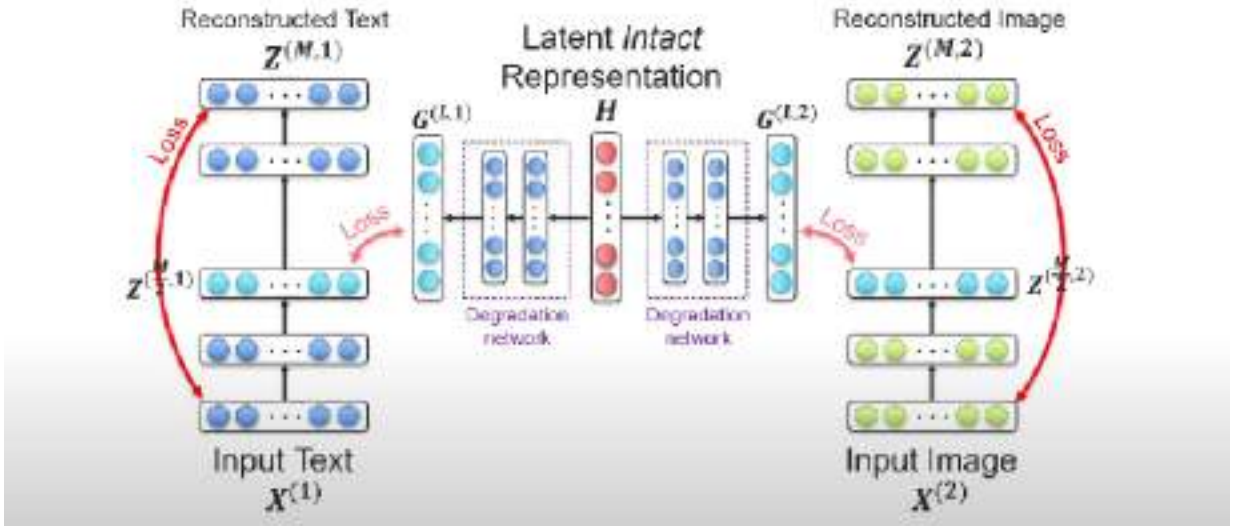


1) There is an "intact" representation which is *complete* and *not damaged*
2) The views z_i are partial (and possibly degenerated) representations of the intact representation

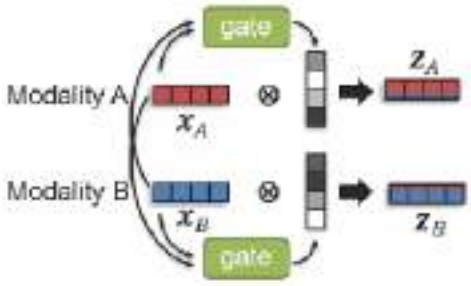
Auto-Encoder in Auto-Encoder Network



Auto-Encoder in Auto-Encoder Network



Gated Coordination



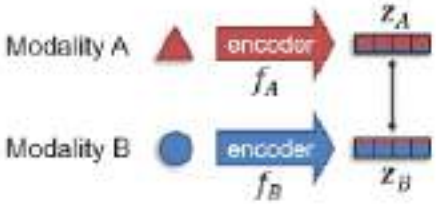
Gated coordination:

$$z_A = g_A(x_A, x_B) \cdot x_A$$

$$z_B = g_B(x_A, x_B) \cdot x_B$$

⟹ Related to attention modules in transformers

Coordination with Contrastive Learning



Paired data: {▲, ●}

(e.g., images and text descriptions)

Positive pairs

Negative pairs

Contrastive loss:

➡ brings positive pairs closer and pushes negative pairs apart

Simple contrastive loss:

$$\max\{0, \alpha + sim(z_A, z_B^-) - sim(z_A, z_B^+)\}$$

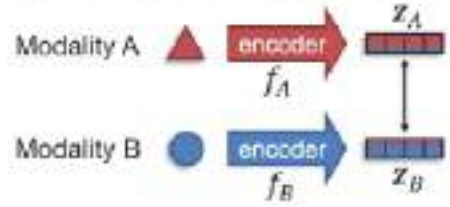
positive pairs

negative pair

Similarity functions are often cosine similarity

➡ Similar to hinge loss

Coordination with Contrastive Learning



Paired data: {▲, ●}

(e.g., images and text descriptions)

Positive pairs

Negative pairs

Contrastive loss:

➡ brings positive pairs closer and pushes negative pairs apart

Simple contrastive loss:

$$\max\{0, \alpha + sim(z_A, z_B^-) - sim(z_A, z_B^+)\}$$

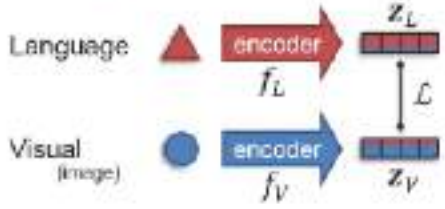
positive pairs

negative pair

Similarity functions are often cosine similarity

➡ Similar to hinge loss

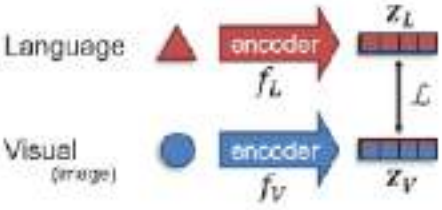
Example – Visual-Semantic Embeddings



Two contrastive loss terms:

$$\max\{0, \alpha + sim(\mathbf{z}_L, \mathbf{z}_V^+) - sim(\mathbf{z}_L, \mathbf{z}_V^-)\}$$
$$+ \max\{0, \alpha + sim(\mathbf{z}_V, \mathbf{z}_L^+) - sim(\mathbf{z}_V, \mathbf{z}_L^-)\}$$

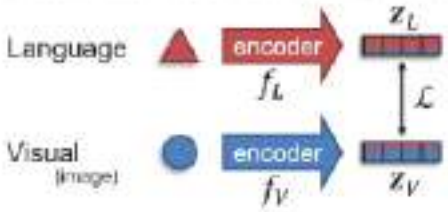
Example – Visual-Semantic Embeddings



Two contrastive loss terms:

$$\max\{0, \alpha + sim(\mathbf{z}_L, \mathbf{z}_V^+) - sim(\mathbf{z}_L, \mathbf{z}_V^-)\}$$
$$+ \max\{0, \alpha + sim(\mathbf{z}_V, \mathbf{z}_L^+) - sim(\mathbf{z}_V, \mathbf{z}_L^-)\}$$

Example – CLIP (Contrastive Language–Image Pre-training)



Popular contrastive loss: InfoNCE

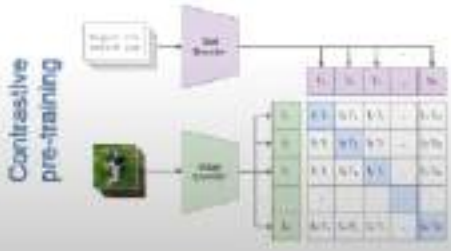
$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} log \frac{sim(z_A^i, z_B^i)}{\sum_{j=1}^{N} sim(z_A^i, z_B^j)}$$

positive pairs

Similarity function can be cosine similarity

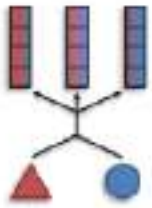
negative pairs and positive pairs

Positive and negative pairs:



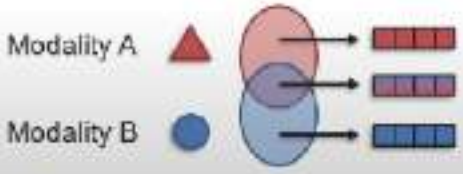
- CLIP encoders (f_L and f_V) are great for language-vision tasks
- z_L and z_V are coordinated but not identical representation spaces

Sub-Challenge 1c: Representation Fission



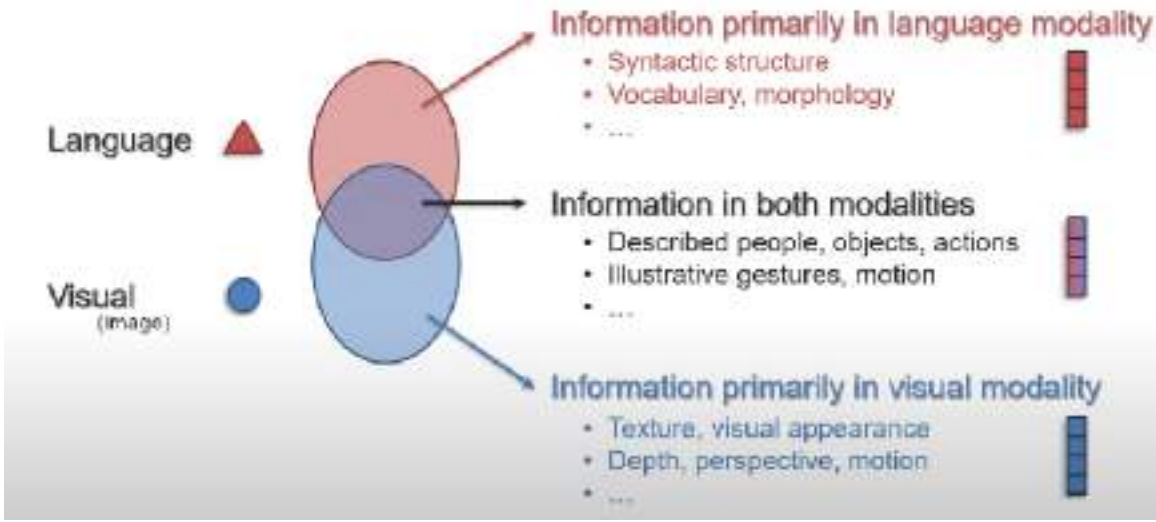
Definition: learning a new set of representations that reflects multimodal internal structure such as data factorization or clustering

Modality-level fission:

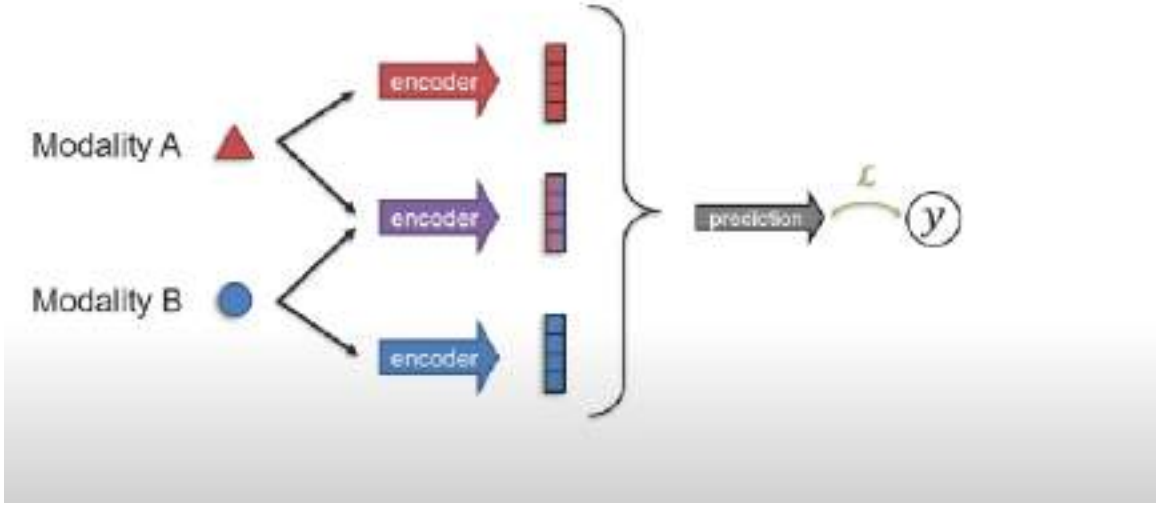


Fine-grained fission:

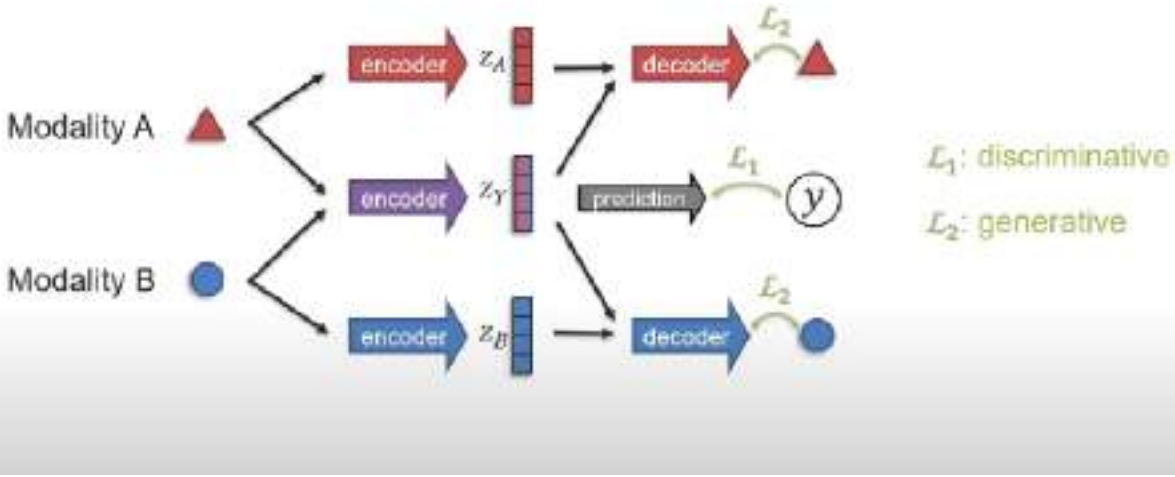
Modality-Level Fission



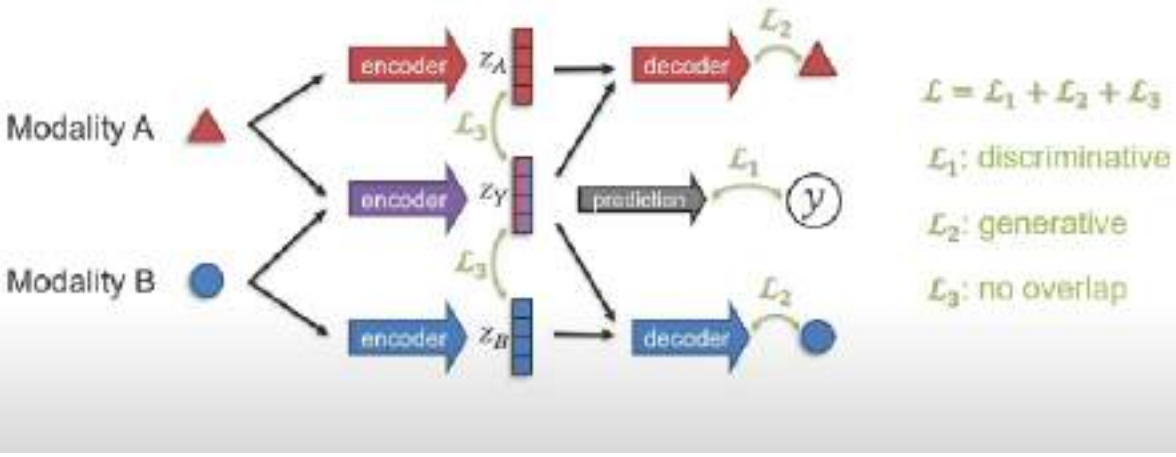
A Discriminative Approach – Factorized Multimodal Representations



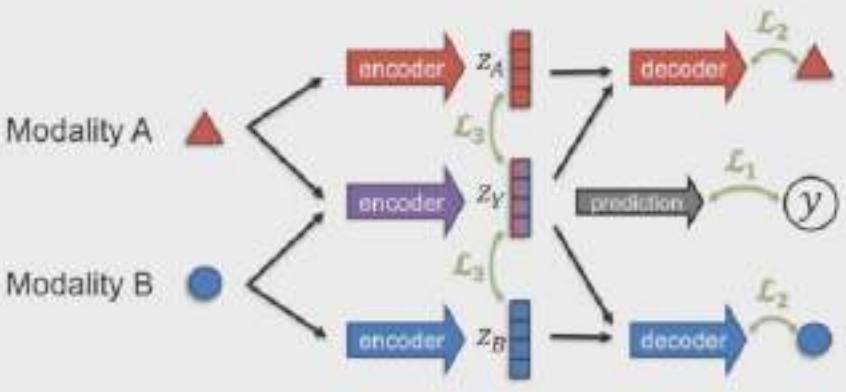
A Generative-Discriminative Approach



A Generative-Discriminative Approach



A Generative-Discriminative Approach



$\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2 + \mathcal{L}_3$

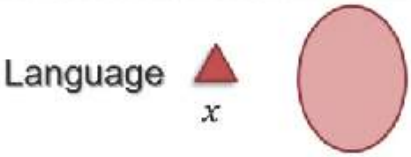
$\mathcal{L}_1$: discriminative

$\mathcal{L}_2$: generative

$\mathcal{L}_3$: no overlap

Separate priors for z_A, z_B and z_Y

Information and Entropy – Information Theory



How much information in the modality?

Information Theory (Shannon, 1948)

Main intuition: "Information value" of a communicated message x depends on how surprising its content is

x: "12, 34, 45, 62 was not a winning combination"

⇨ Not surprising… So, low information

x: "11, 28, 38, 58 was a winning combination"

⇨ Low chances… So, higher information

Information content $I(x)$

$$I(x) \sim \frac{1}{p(x)}$$

Information and Entropy – Information Theory

Language x

How much information in the modality?

Information Theory (Shannon, 1948)

Main intuition: "Information value" of a communicated message x depends on how surprising its content is

x: "12, 34, 45, 62 was not a winning combination"

➡ Not surprising… So, low information

x: "11, 28, 38, 58 was a winning combination"

➡ Low chances… So, higher information

Information content $I(x)$

$$I(x) \sim \frac{1}{p(x)}$$ ➡ But how to scale?

$$I(x) = \log\left(\frac{1}{p(x)}\right) = -\log(p(x))$$

Information and Entropy – Information Theory

Language x

How much information in the modality?

Information Theory (Shannon, 1948)

Information content $I(X) = -\log(p(X))$

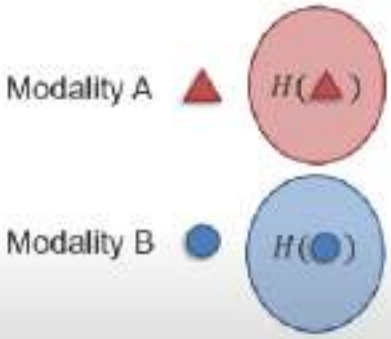
➡ For discrete alphabet $\mathcal{X}$, then X is discrete random variable

Entropy: weighted average of all possible outcomes from $\mathcal{X}$

$$H(X) = \mathbb{E}[I(X)] = \mathbb{E}[-\log(p(X))] = -\sum_{x \in \mathcal{X}} p(X)\log(p(X))$$

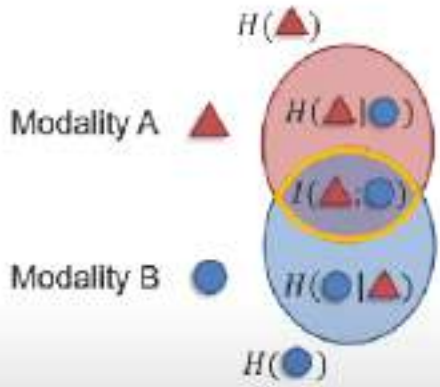
Entropy with Two Modalities

If no overlapping information



But in most real-world scenarios, modalities are *inter-connected*

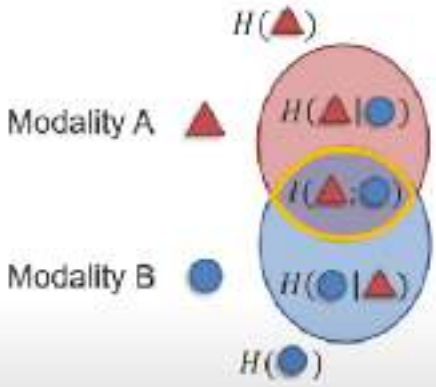
Entropy with Two Modalities



Mutual information $I(X;Y)$

$$I(X;Y) = H(X) - H(X|Y)$$

Entropy with Two Modalities

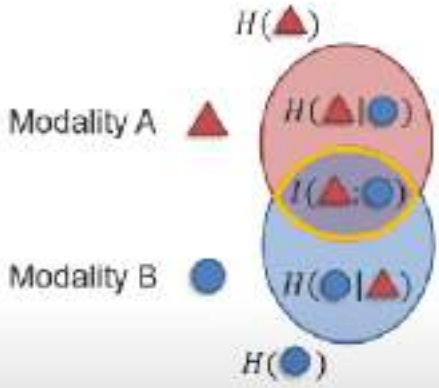


Mutual information $I(X;Y)$

$$I(X;Y) = H(X) - H(X|Y)$$

$$= \mathbb{E}_{X,Y}\left[\log\frac{1}{P_X(x)} + \log\frac{P_{XY}(x,y)}{P_Y(y)}\right]$$

Entropy with Two Modalities

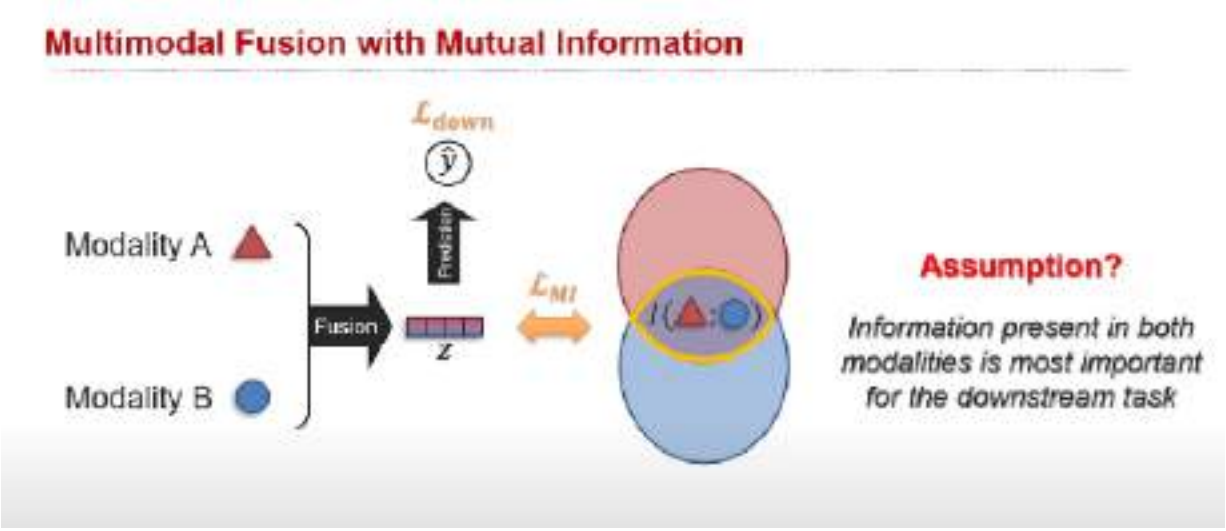


Mutual information $I(X;Y)$

$$I(X;Y) = H(X) - H(X|Y)$$

$$= \mathbb{E}_{X,Y}\left[\log\frac{1}{P_X(x)} + \log\frac{P_{XY}(x,y)}{P_Y(y)}\right]$$

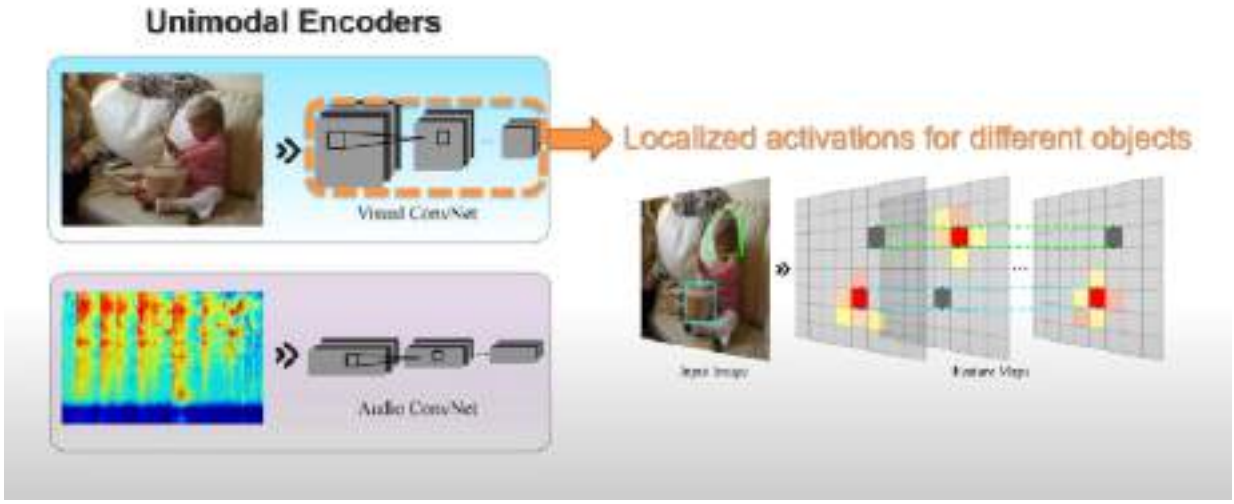
$$I(X;Y) = \mathbb{E}_{X,Y}\left[\log\frac{P_{XY}(x,y)}{P_X(x)P_Y(y)}\right]$$


Multimodal Fusion with Mutual Information
$\mathcal{L}_{down}$
$\hat{y}$
Predictor
Modality A
Modality B
Fusion
z
$\mathcal{L}_{MI}$
$I(\triangle;\bigcirc)$
Assumption?
Information present in both modalities is most important for the downstream task

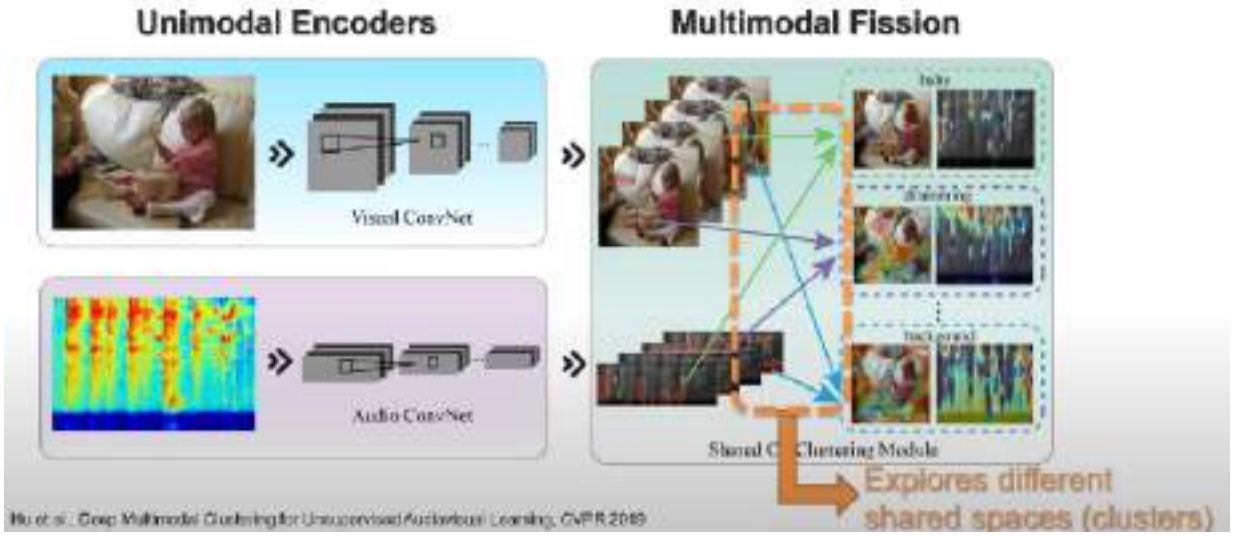
---

Lecture 4.2

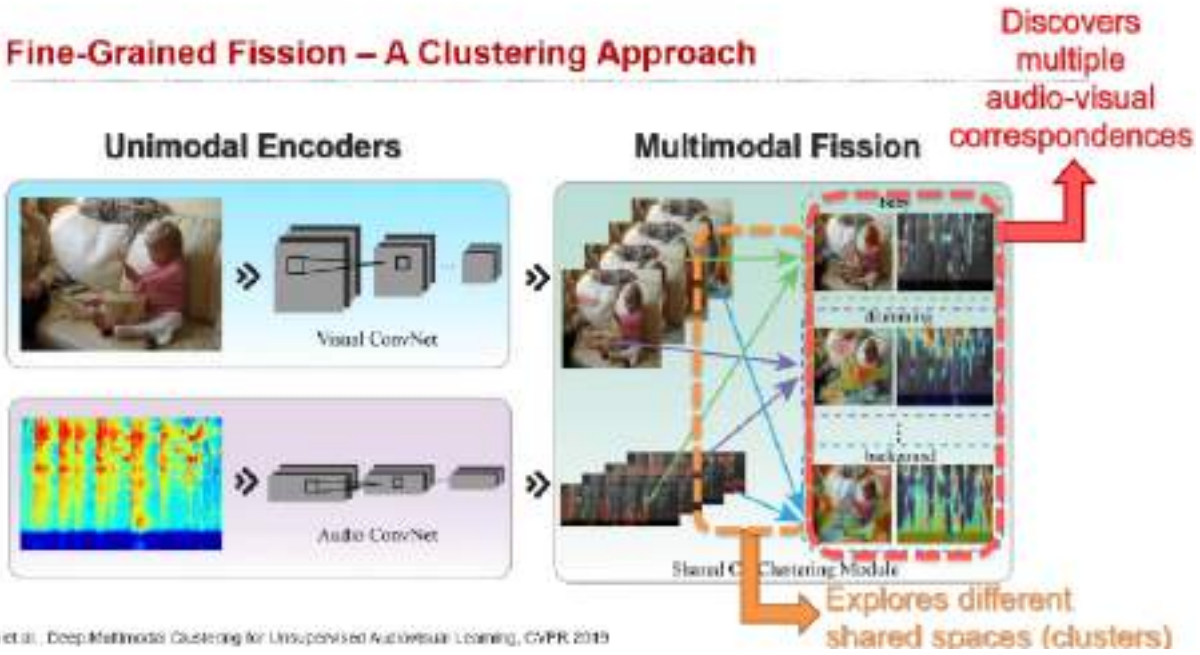
Fine-Grained Fission – A Clustering Approach



Fine-Grained Fission – A Clustering Approach



Fine-Grained Fission – A Clustering Approach



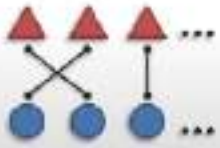
Hu et al. Deep Multimodal Clustering for Unsupervised Audiovisual Learning, CVPR 2019

Challenge 2: Alignment

Definition: Identifying and modeling cross-modal connections between all elements of multiple modalities, building from the data structure

Sub-challenges:

Discrete Alignment



Discrete elements and connections

Continuous Alignment

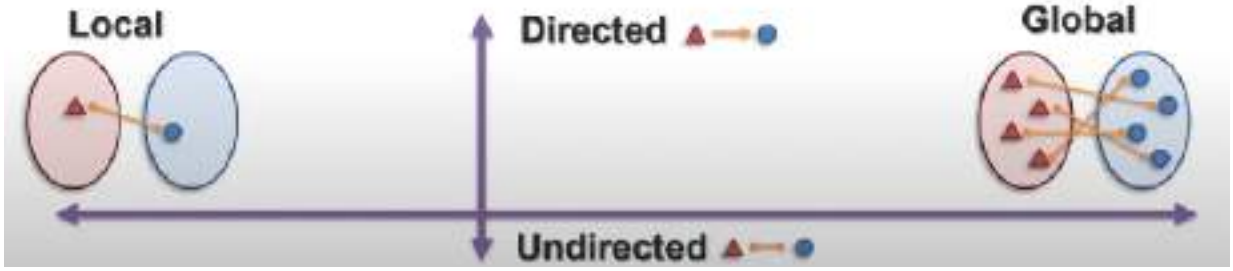
Segmentation and continuous warping

Contextualized Representation

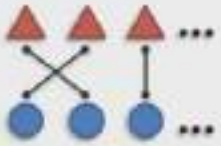
Alignment + representation

Sub-Challenge 2a: Discrete Alignment

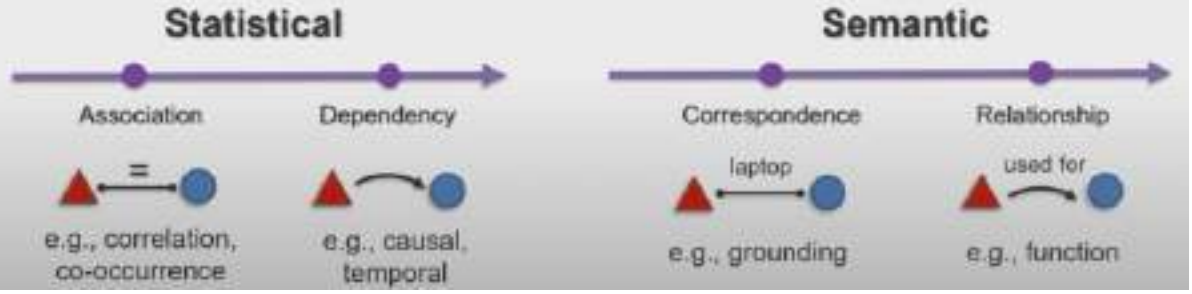
Definition: Identify and model connections between elements of multiple modalities



Connections



Why should 2 elements be connected?



Connections

Why should 2 elements be connected?

Relationships and Dependencies will be discussed in more details in **Reasoning** challenge

Language Grounding

Definition: Tying language (words, phrases,…) to non-linguistic elements, such as the visual world (objects, people, …)

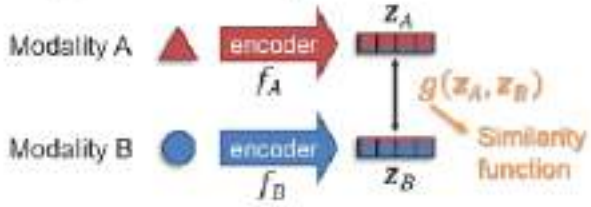
Local Alignment – Coordinated Representations

Visual

Language

A woman reading newspaper

Learning coordinated representations:



Supervision: Paired data

Local Alignment – Coordinated Representations

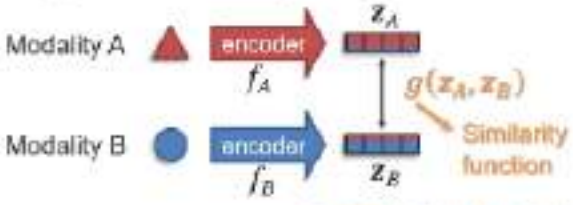
Visual

Language

Common information

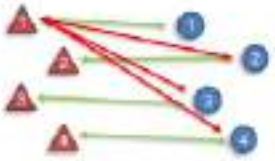
A woman reading newspaper

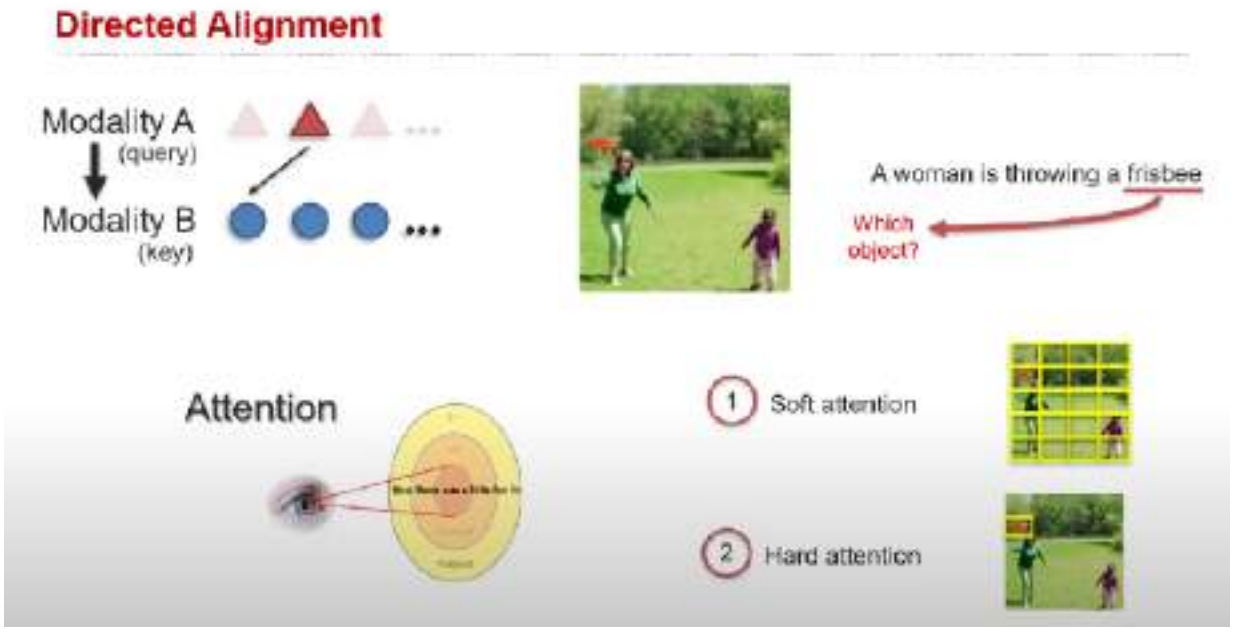
Learning coordinated representations:

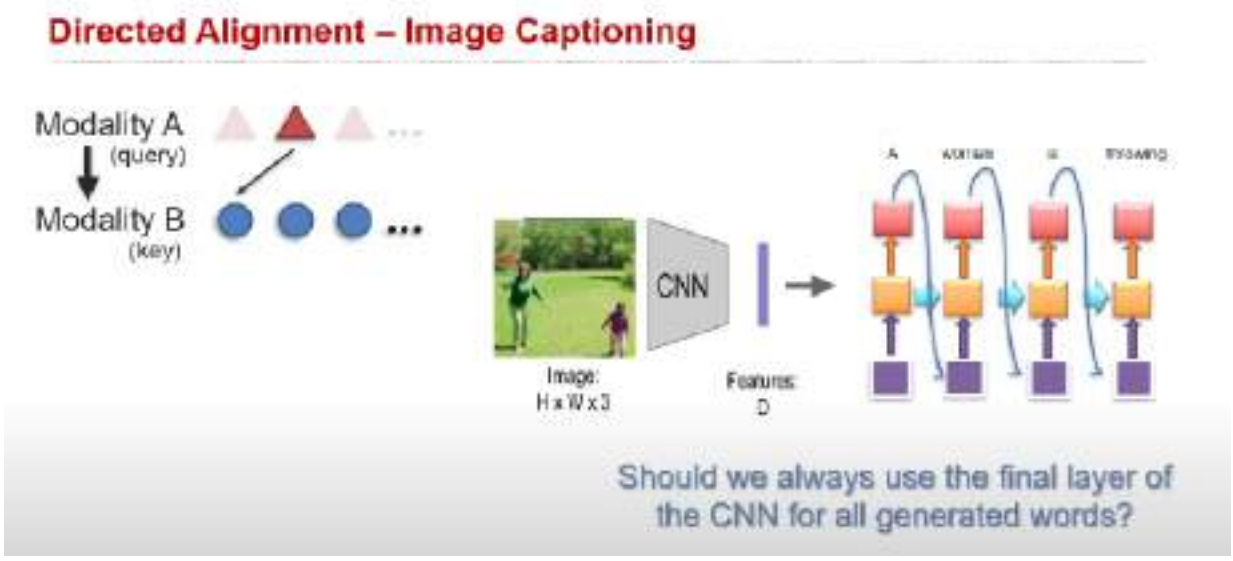


or contrastive learning

Supervision: Paired data

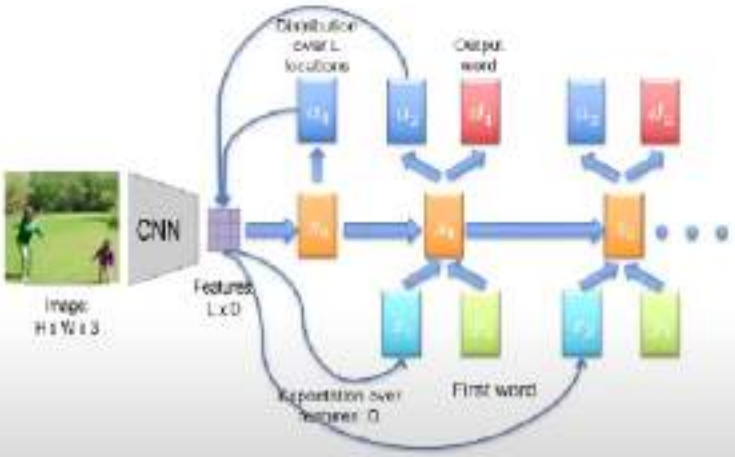






Need to understand more about attention

Directed Alignment – Image Captioning



Attention Gates

Before:

$$p(y_i|y_1, \ldots, y_{i-1}, \boldsymbol{x}) = g(y_{i-1}, \boldsymbol{s}_i, \boldsymbol{z}),$$

where $\boldsymbol{z} = \boldsymbol{h}_T$, last encoder state and $\boldsymbol{s}_i$ is the current state of the decoder

Now:

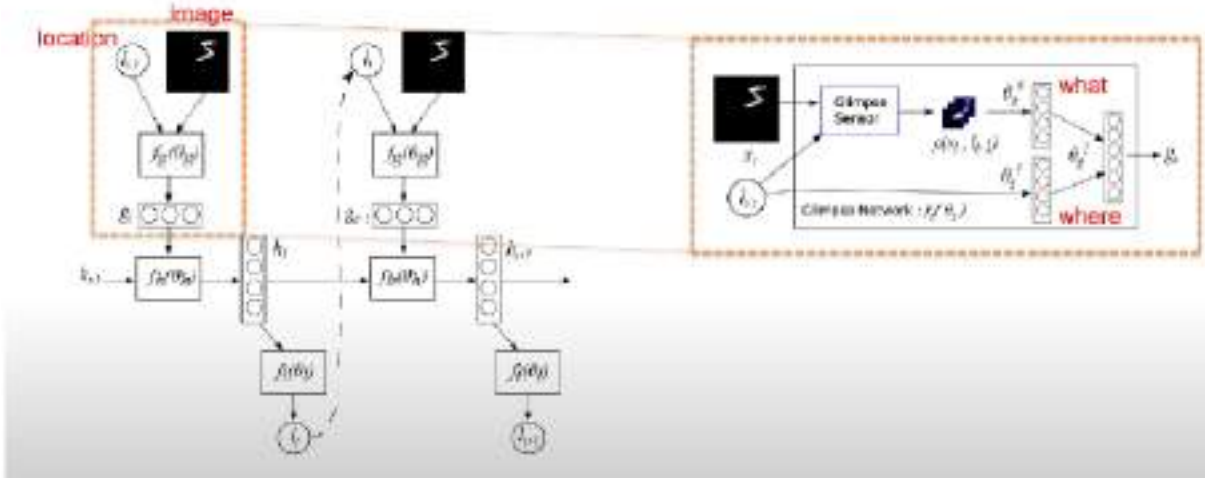
$$p(y_i|y_1, \ldots, y_{i-1}, \boldsymbol{x}) = g(y_{i-1}, \boldsymbol{s}_i, \boldsymbol{z}_i)$$

Have an attention "gate"

- A different context $\boldsymbol{z}_i$ used at each time step!
- $\boldsymbol{z}_i = \sum_{j=1}^{T_x} \alpha_{ij} \boldsymbol{h}_j$

α_{ij} is the (scalar) attention for word j at generation step i

Hard Attention – Recurrent Model of Visual Attention



RNN Tasks

Image Captioning

A woman is throwing a frisbee in a park

Neural Machine Translation

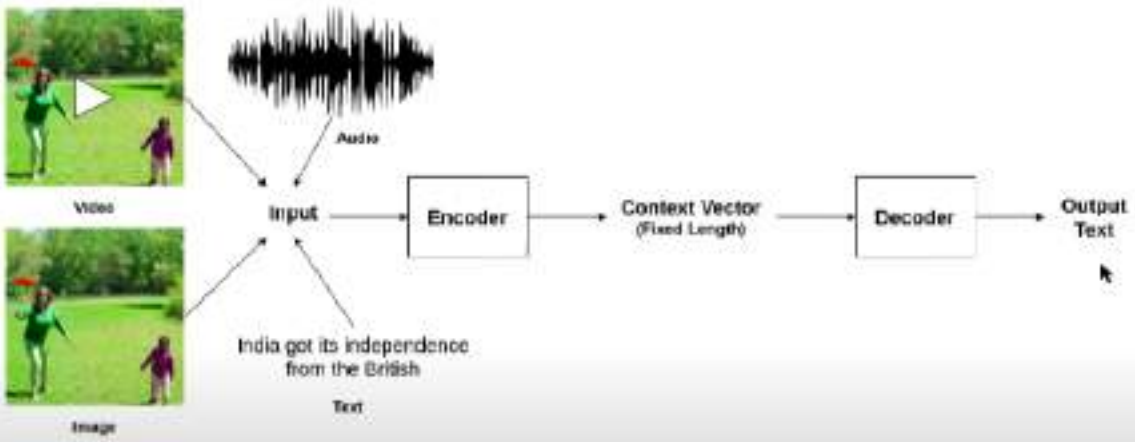
India got its independence from the British

भारत को अंग्रेजों से आजादी मिली

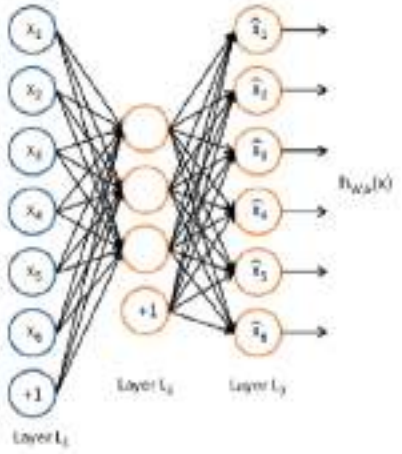
L'accord sur l'Espace économique européen a été signé en août 1992

The agreement on the European Economic Area was signed in August 1992

Encoder-Decoder Modeling

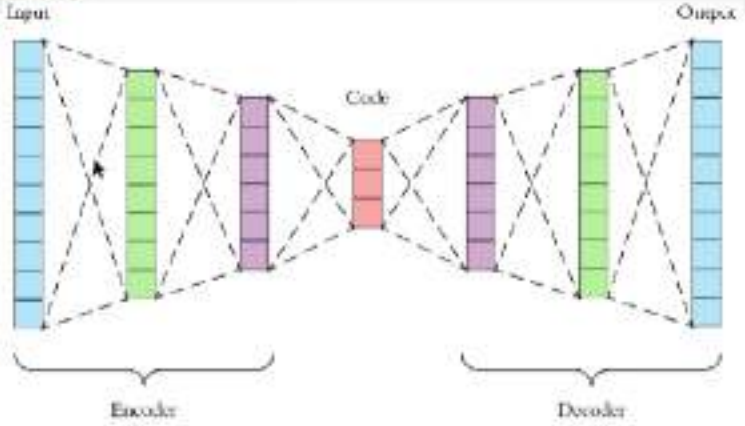


Autoencoders



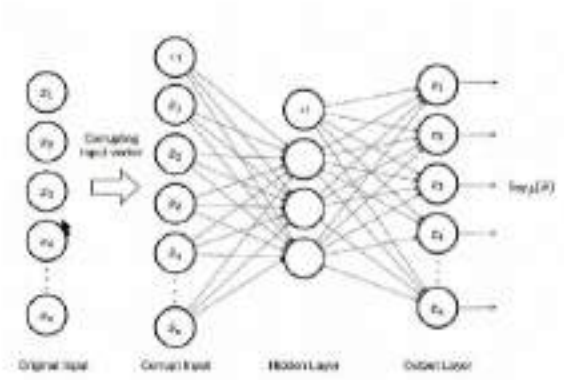
- An **autoencoder neural network** is an unsupervised learning model that applies backpropagation, setting target values to be equal to inputs themselves, i.e. $\mathbf{y}_i = \mathbf{x}_i$
- Learns a function $f_{W,b}(\mathbf{x}) \approx \mathbf{x}$; in other words, learn an approximation to identity function, output $\hat{\mathbf{x}}$ close to $\mathbf{x}$
- Loss function? Mean Squared Error $\mathcal{L} = ||\mathbf{x} - \hat{\mathbf{x}}||_2^2$

Deep Autoencoders



Both encoder and decoder can have many layers too - in standard deep autoencoders, the architecture in encoder is often mirrored in decoder (not always so though)

Denoising Autoencoders



- Variant of autoencoder, where input is perturbed with noise (e.g. Gaussian), but network is asked to predict original input without noise
- Loss function? Mean Squared Error between output and original uncorrupted input

More on Autoencoders

Why should the hidden layers be smaller in size than input layer?

- Autoencoder (AE) with hidden layer with lesser dimension than input layer called **undercomplete AE** $\Longrightarrow$ AE learns a lower-dimensional representation on suitable manifold of input data

More on Autoencoders

Why should the hidden layers be smaller in size than input layer?

- Autoencoder (AE) with hidden layer with lesser dimension than input layer called **undercomplete AE** $\implies$ AE learns a lower-dimensional representation on suitable manifold of input data
- Autoencoder (AE) with hidden layer with higher dimension than input layer called **overcomplete AE** $\implies$ AE could learn trivial solutions, by copying input!

More on Autoencoders

Why should the hidden layers be smaller in size than input layer?

- Autoencoder (AE) with hidden layer with lesser dimension than input layer called **undercomplete AE** $\implies$ AE learns a lower-dimensional representation on suitable manifold of input data
- Autoencoder (AE) with hidden layer with higher dimension than input layer called **overcomplete AE** $\implies$ AE could learn trivial solutions, by copying input!

Are autoencoders then dimensionality reduction methods?

- Yes, indeed; undercomplete AEs are dimensionality reduction methods

Back to NMT Model (Seq2Seq Model)

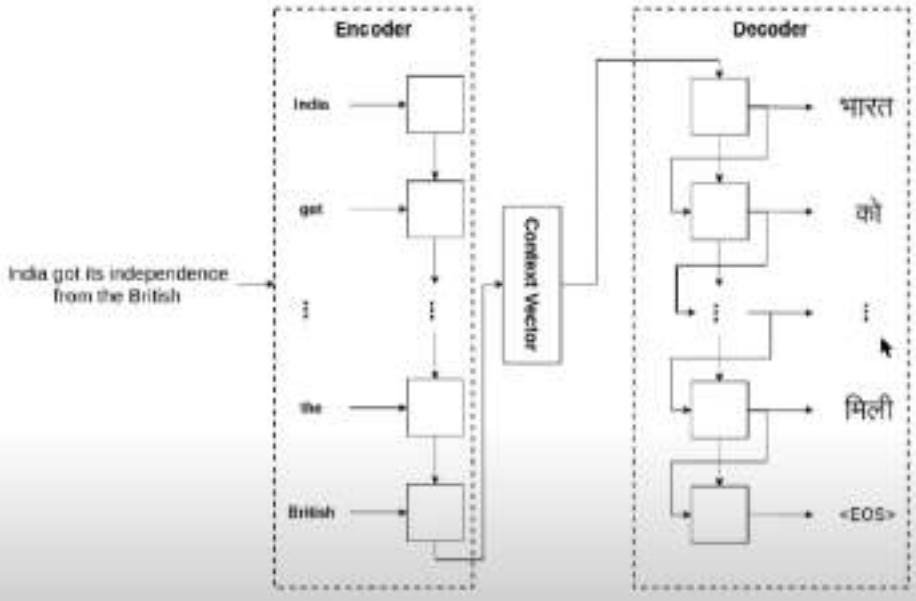
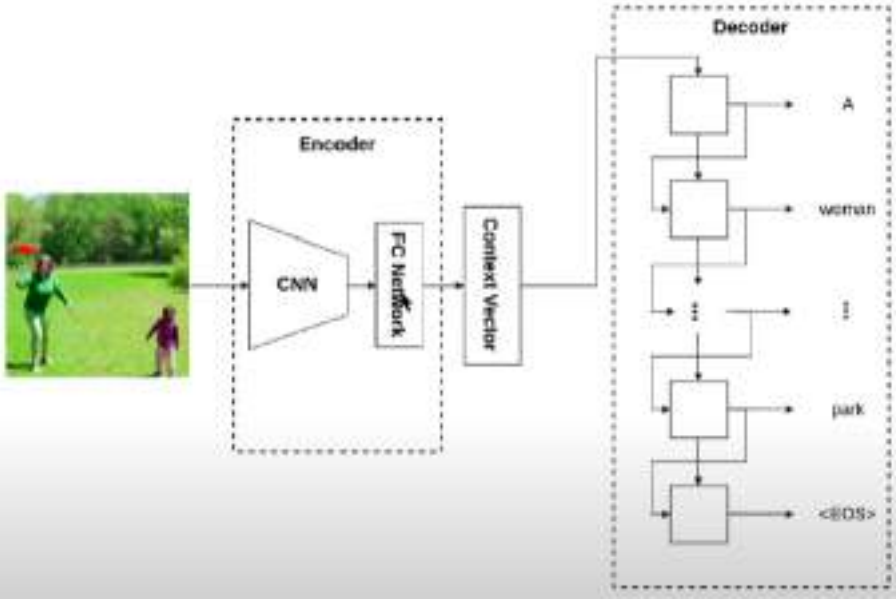
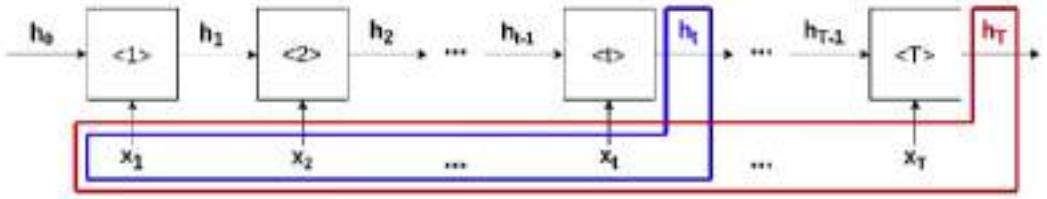


Image Captioning Model



What is the problem?



A hidden state at time step t (h_t) is a compressed form of all previous inputs $(x_1, x_2, ..., x_t)$

What is the problem?

After meeting a comrade at the last post station but one before Moscow, Denisov had drunk three bottles of wine with him and, despite the jolting ruts across the snow-covered road, did not once wake up on the way to Moscow, but lay at the bottom of the sleigh beside Rostov, who grew more and more impatient the nearer they got to Moscow. <EOS>

Excerpts from the work of Leo Tolstoy(~60 words)

But what if input is very long? Can h_T encode all information without forgetting? Information bottleneck!

What is the problem?

Question: What is the name of the book?
Answer: The name of the book is Lord of the Rings.

Credit: Bharath Kishore, Flickr CC License

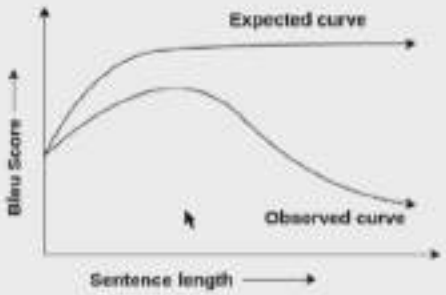
Visual Question Answering (VQA)
(To be discussed later)

Relevant information in a cluttered image should also be preserved

Failure of Encoder-Decoder Modeling

BLEU score:

- Stands for Bilingual Evaluation Understudy
- Metric for evaluating quality of machine translated text
- Can be used for other language tasks (like Image captioning, VQA, etc)
- For more information, see BLEU score, Wikipedia



Solution?

We can describe attention with the following example :

Attention: Intuition

How do you answer this?

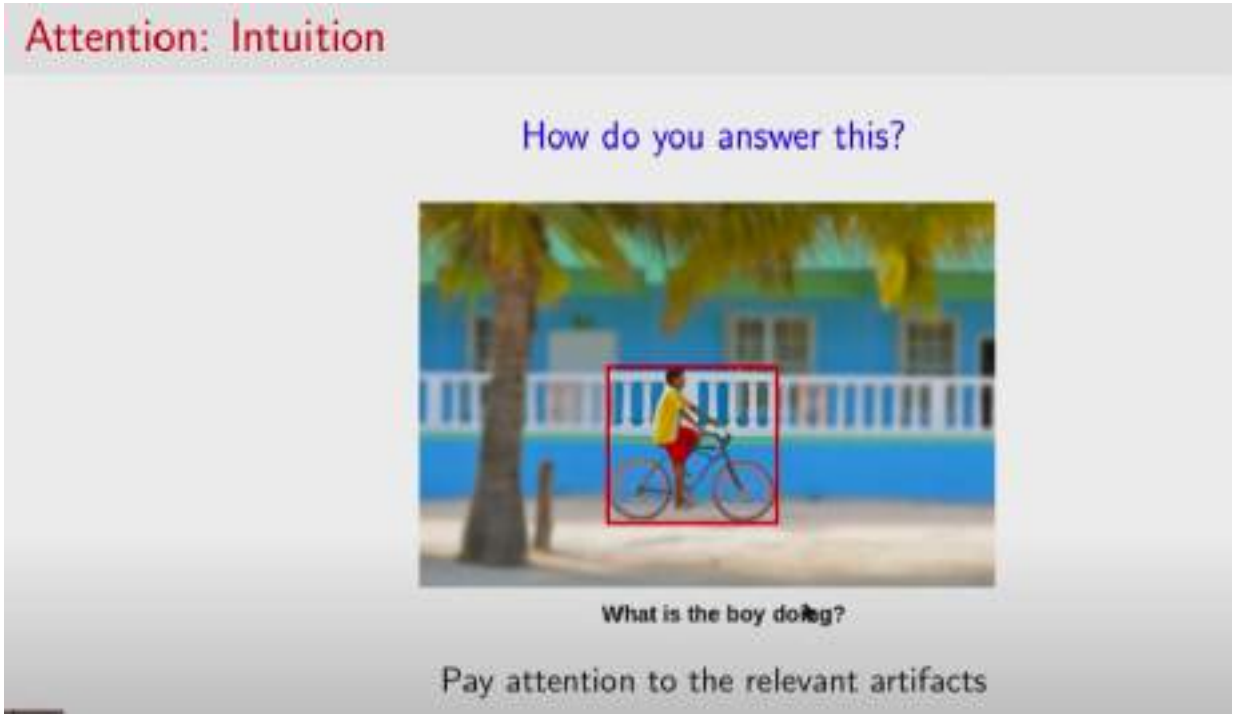
What is the boy doing?

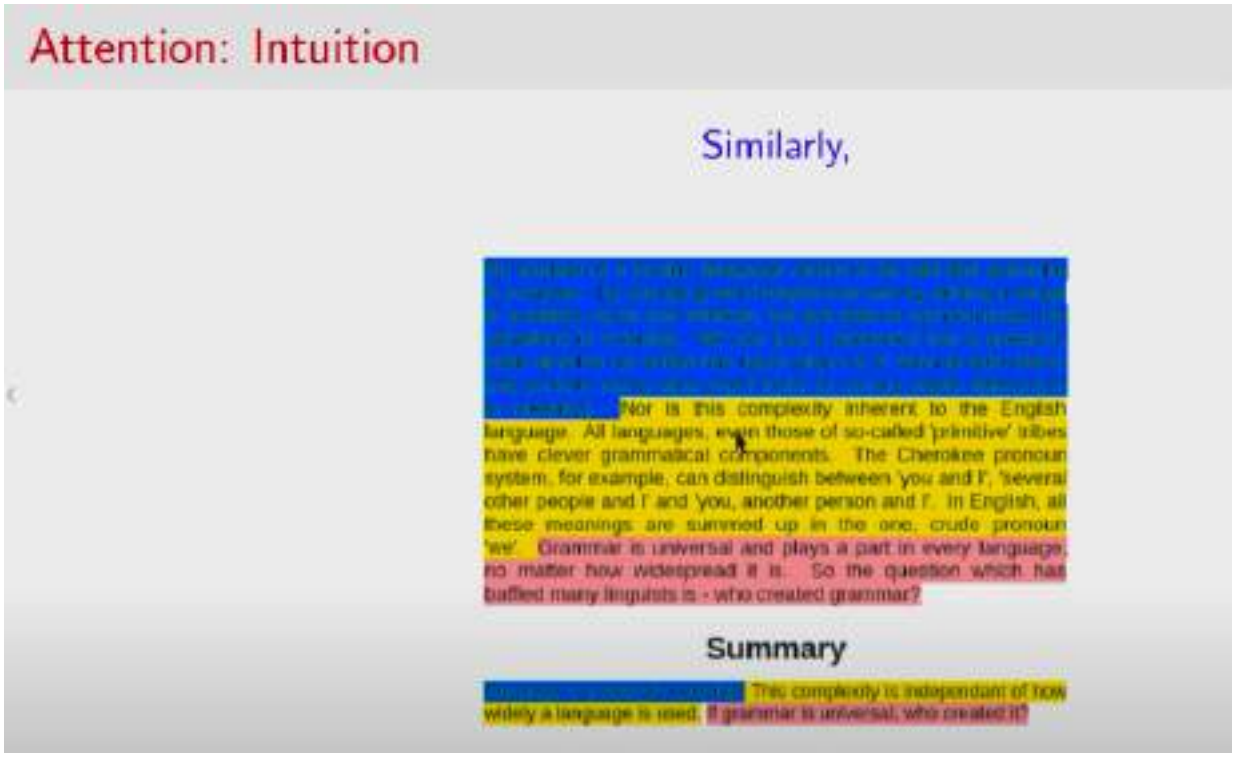
Attention: Intuition

How do you answer this?

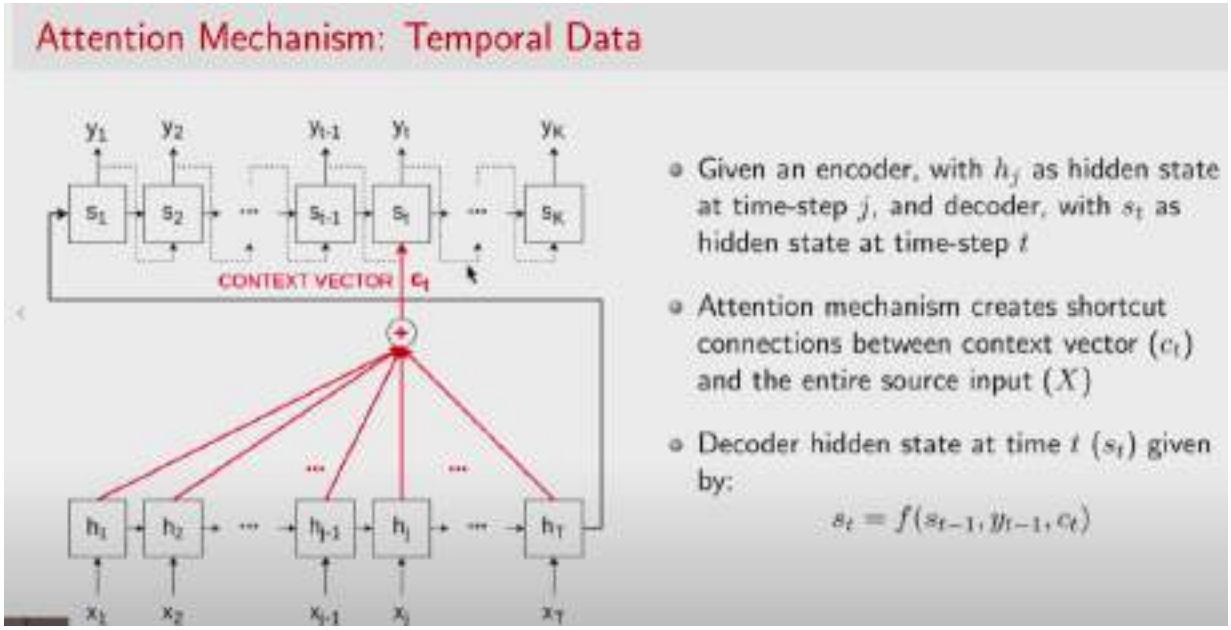
What is the boy doing?

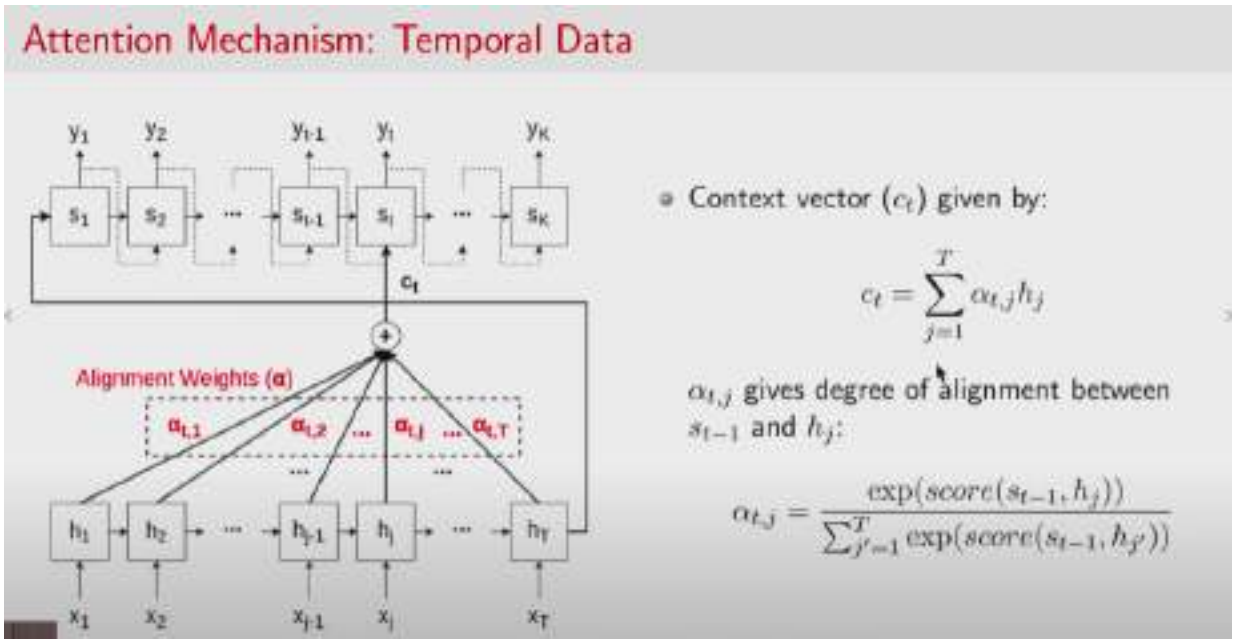
Identify artifacts in the image





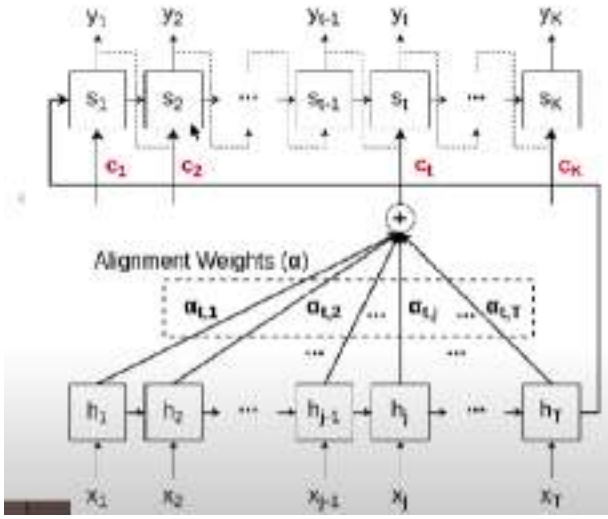
Attention:We are looking at each part of the input and the summarising it





Score basically tells us how much relation is there betweeb s_t-1(current output) with each input

Attention Mechanism: Temporal Data



- Context vector (c_t) given by:

$$c_t = \sum_{j=1}^{T} \alpha_{t,j} h_j$$

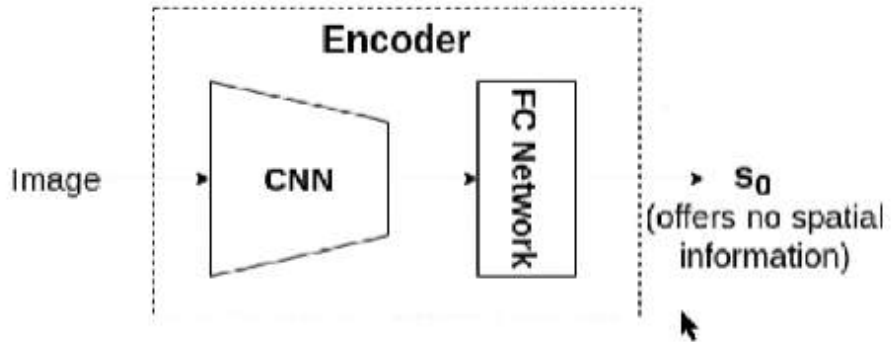
$\alpha_{t,j}$ gives degree of alignment between s_{t-1} and h_j:

$$\alpha_{t,j} = \frac{\exp(score(s_{t-1}, h_j))}{\sum_{j'=1}^{T} \exp(score(s_{t-1}, h_{j'}))}$$

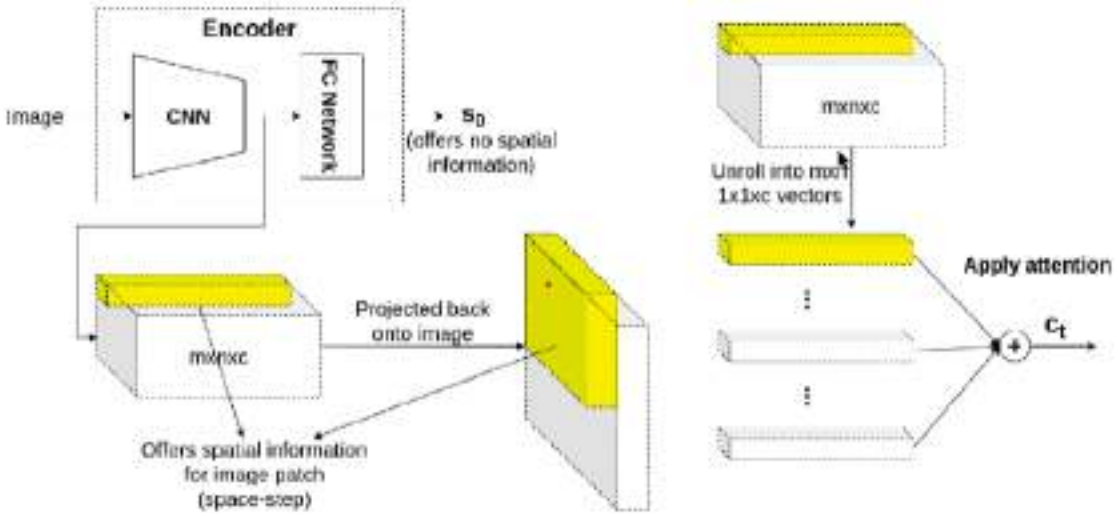
Alignment Scores

Name	Alignment Score Function
Content-based Attention	$\text{score}(s_t, h_i) = \text{cosine}(s_t, h_i)$
Additive Attention	$\text{score}(s_t, h_i) = v_a^T \tanh(W_a[s_t; h_i])$
Location-Based Attention	$\alpha_{t,j} = \text{softmax}(W_a s_t)$
General Attention	$\text{score}(s_t, h_i) = s_t^T W_a h_i$
Dot-Product Attention	$\text{score}(s_t, h_i) = s_t^T h_i$
Scaled Dot-Product Attention	$\text{score}(s_t, h_i) = \frac{s_t^T h_i}{\sqrt{n}}$

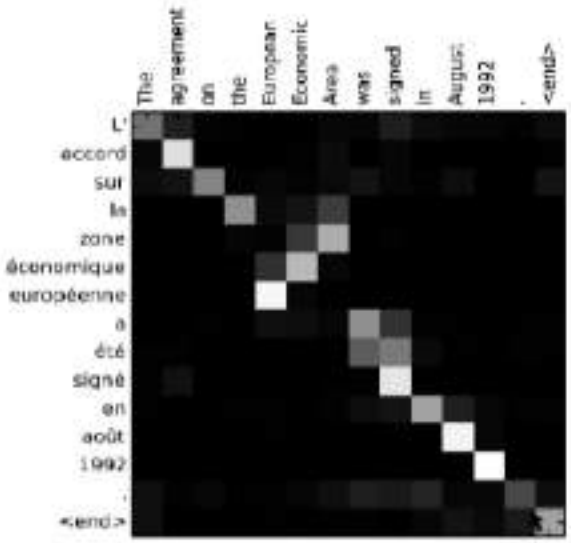
Attention Mechanism: Spatial Data



Attention Mechanism: Spatial Data



Byproduct of Attention: Explainability



English-to-French Translation

Byproduct of Attention: Explainability

Modes of Attention

Hard vs Soft Attention:

- Single position is chosen for full alignment (1.0)
- Position-choosing is stochastic and hence non-differentiable

- All positions get partial alignment weights (0-1)
- Deterministic and hence differentiable, as no position-choosing

Modes of Attention

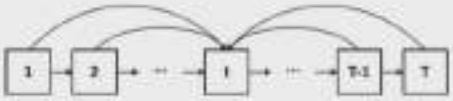
Global vs Local Attention:

All input positions are chosen for attention

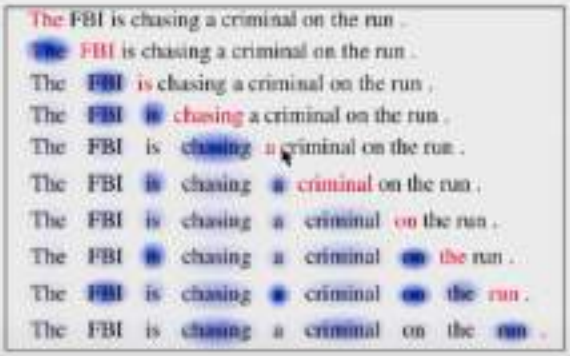
Neighbourhood of aligned position is chosen for attention

Modes of Attention

Self-Attention:



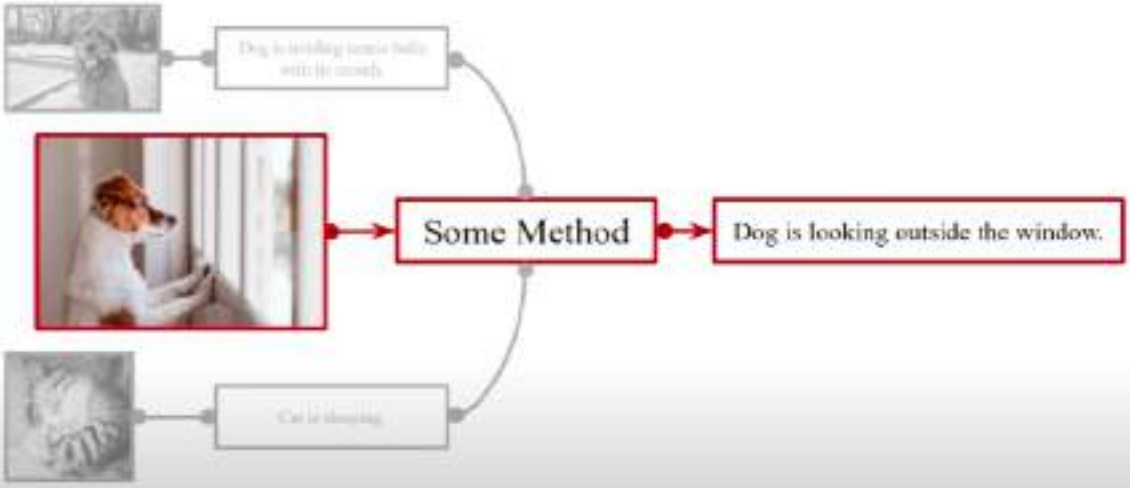
- Also known as intra attention
- Used extensively in advanced attention models (will see more soon)



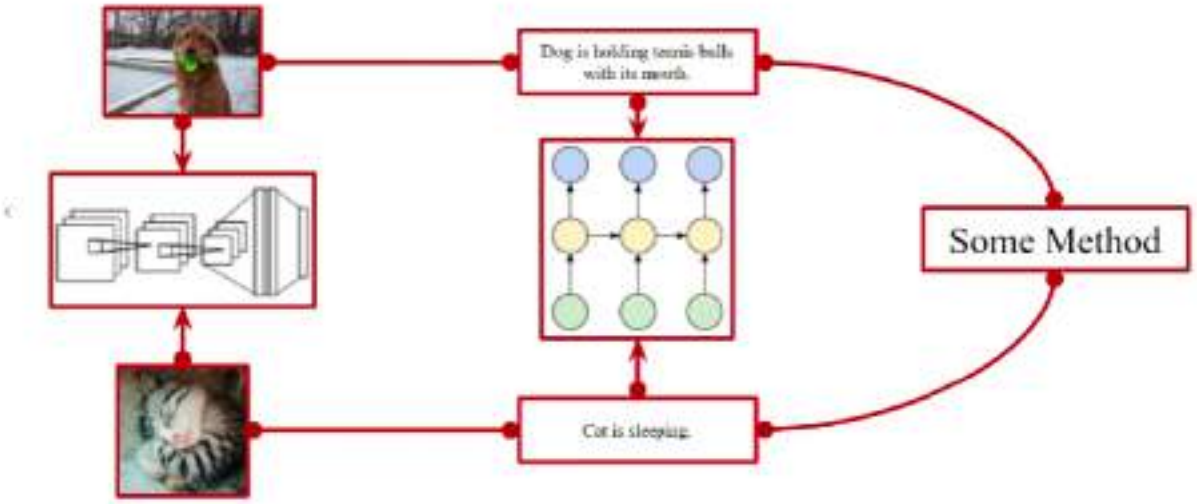
Describe these images

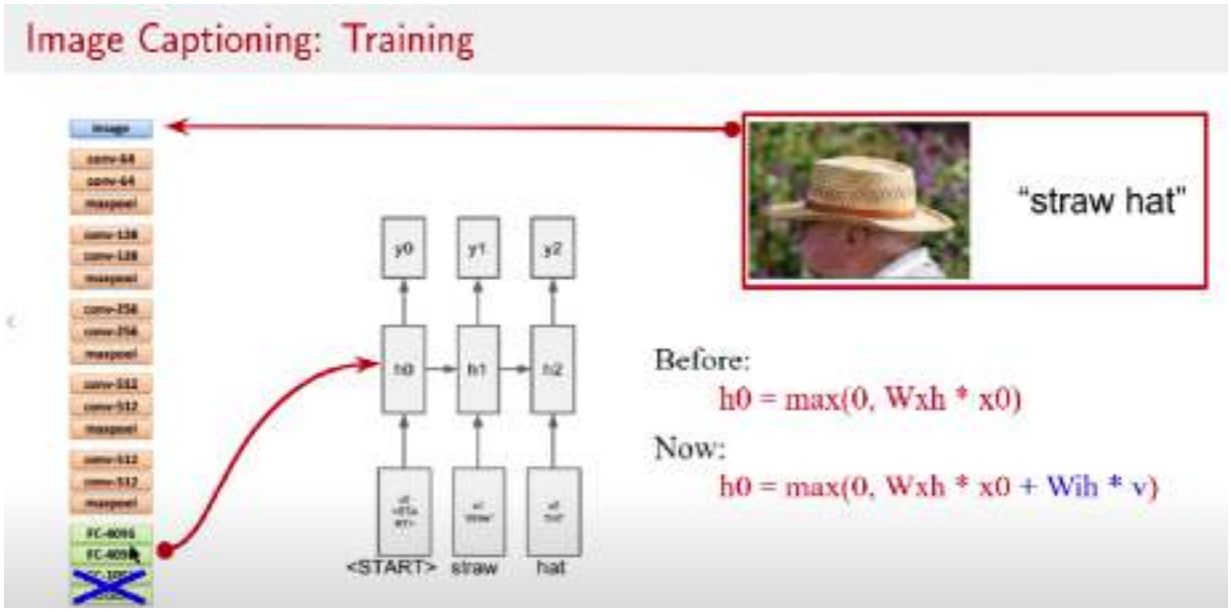
- How can we understand what is happening just by looking at a single image ?
- Can we make a computer do the same?

How to make a computer describe an image?



How to make a computer describe an image?





What happens during testing?

Image Captioning: Inference (Test Time)

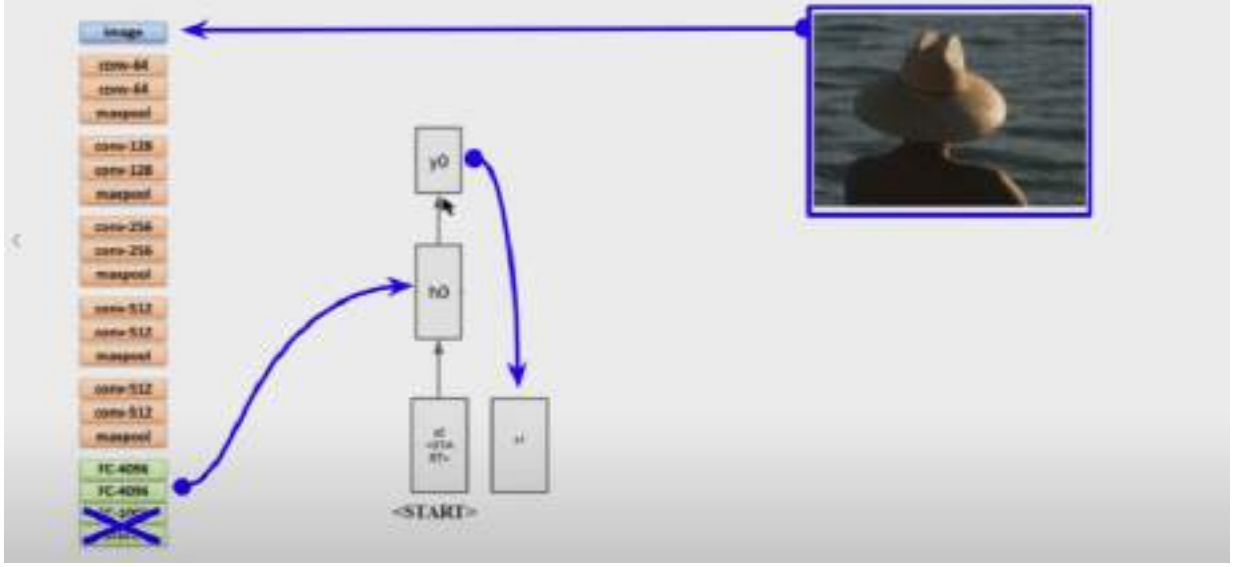
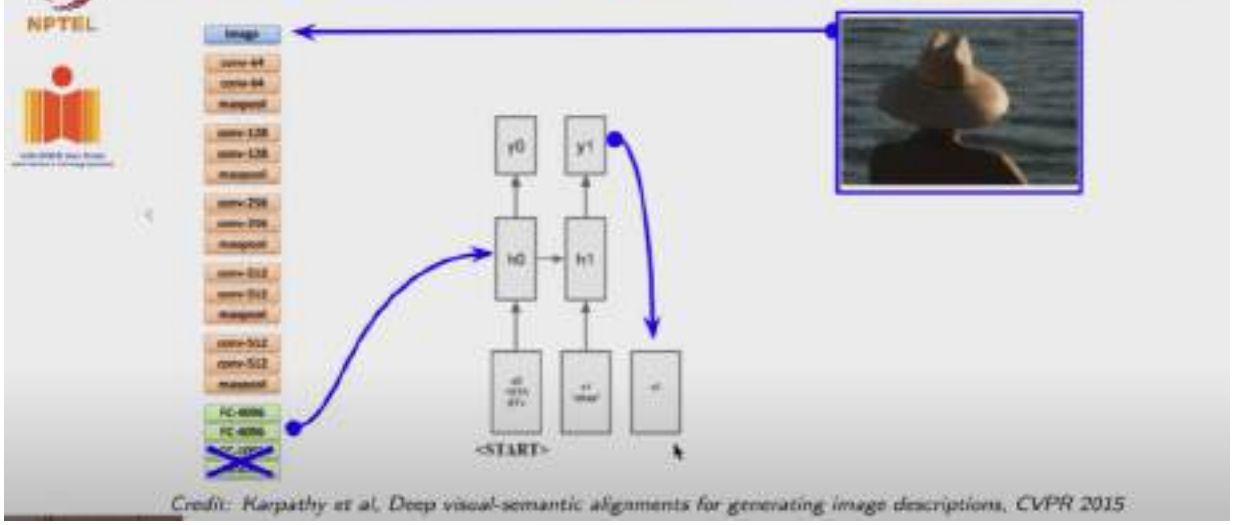
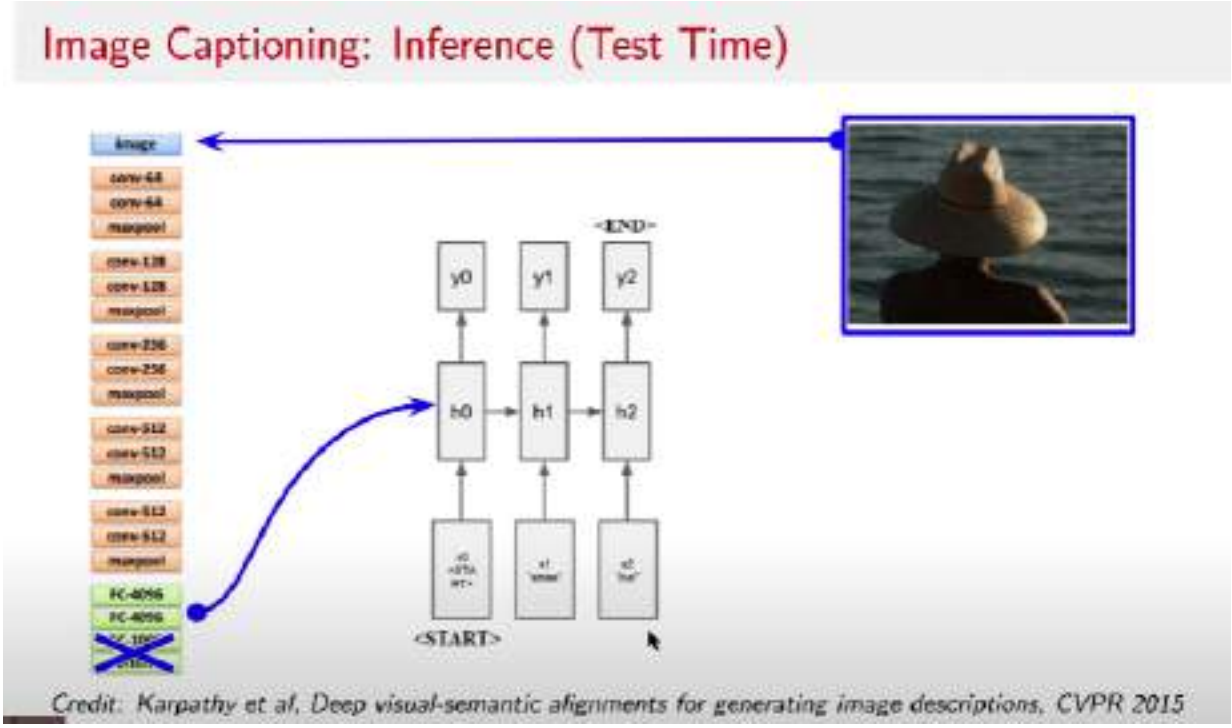


Image Captioning: Inference (Test Time)



Credit: Karpathy et al, Deep visual-semantic alignments for generating image descriptions, CVPR 2015



The start token and end token are treated as additional words and are added to the dictionary ,for example if the no of words in the dictionary are 1000 wit the start and end token it becomes 1002

Results

a group of people standing around a room with remotes
logprob: -9.17

a young boy is holding a baseball bat
logprob: -7.61

a cow is standing in the middle of a street
logprob: -8.84

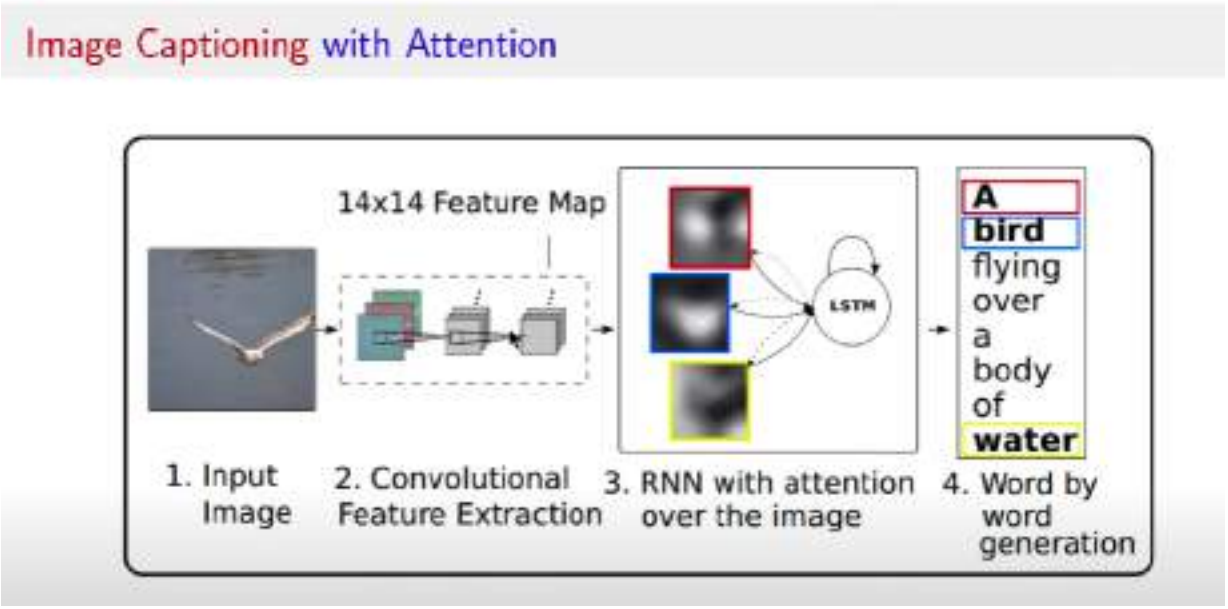
Results: Failure Cases

Possible to understand why the method failed

a man standing next to a clock on a wall
logprob: -10.09

a young boy is holding a baseball bat
logprob: -7.65

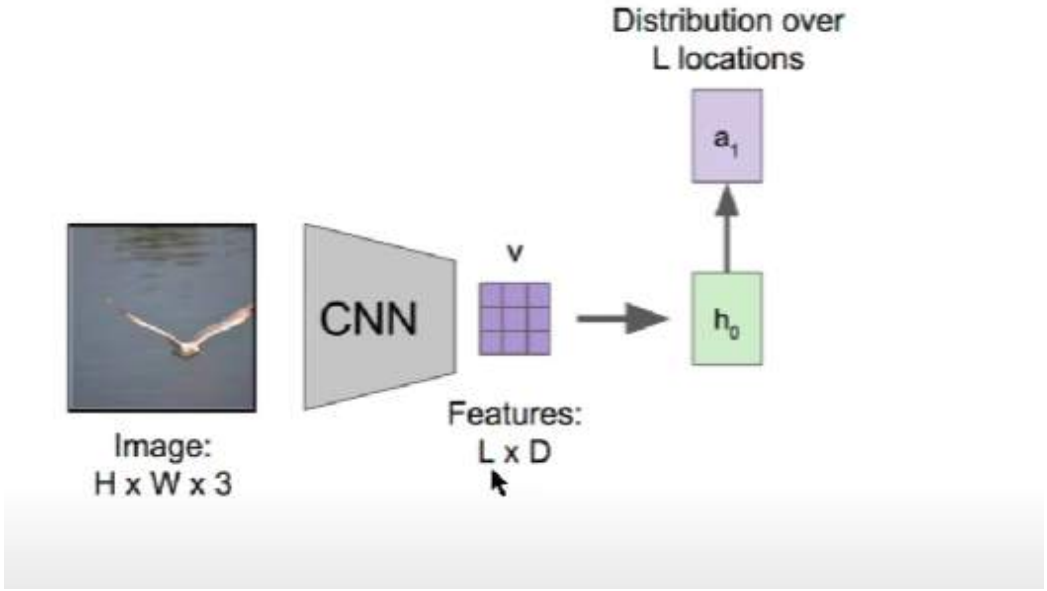
a cat is sitting on a couch with a remote control
logprob: -12.45



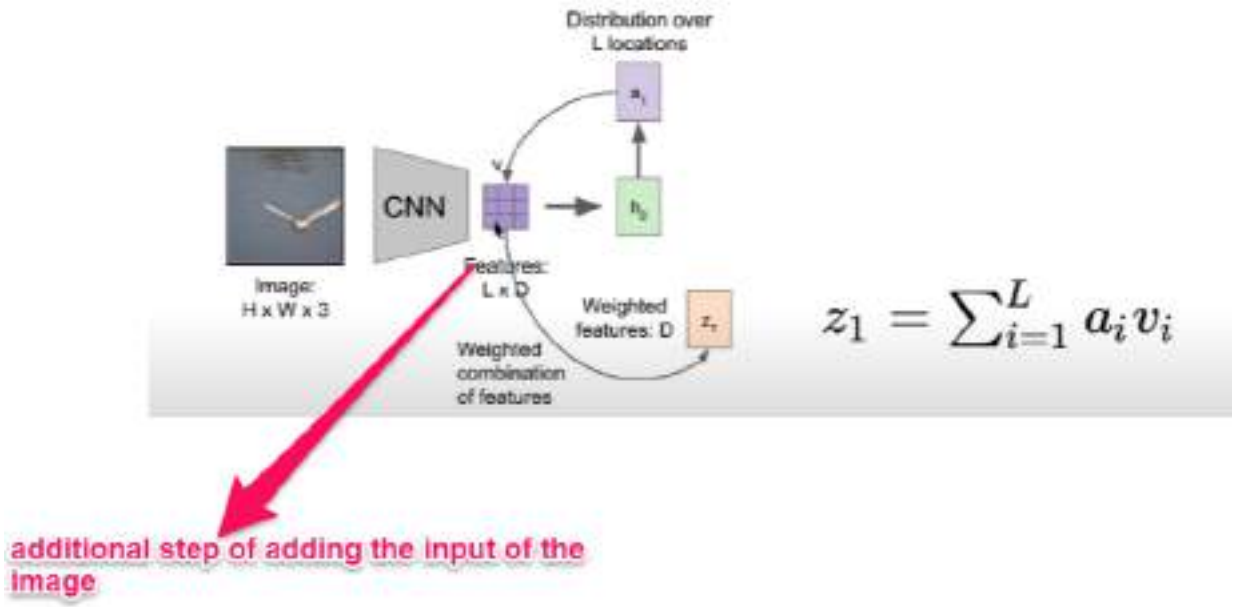
Conventionally RNN outputs a distribution over vocabulry i.e softmax function of the one hot encoded vector but in the case above it is giving a probability over L locations in the image ,locations are the grids in the feature maps,it is using softmax

a1=softmax is length of H*W

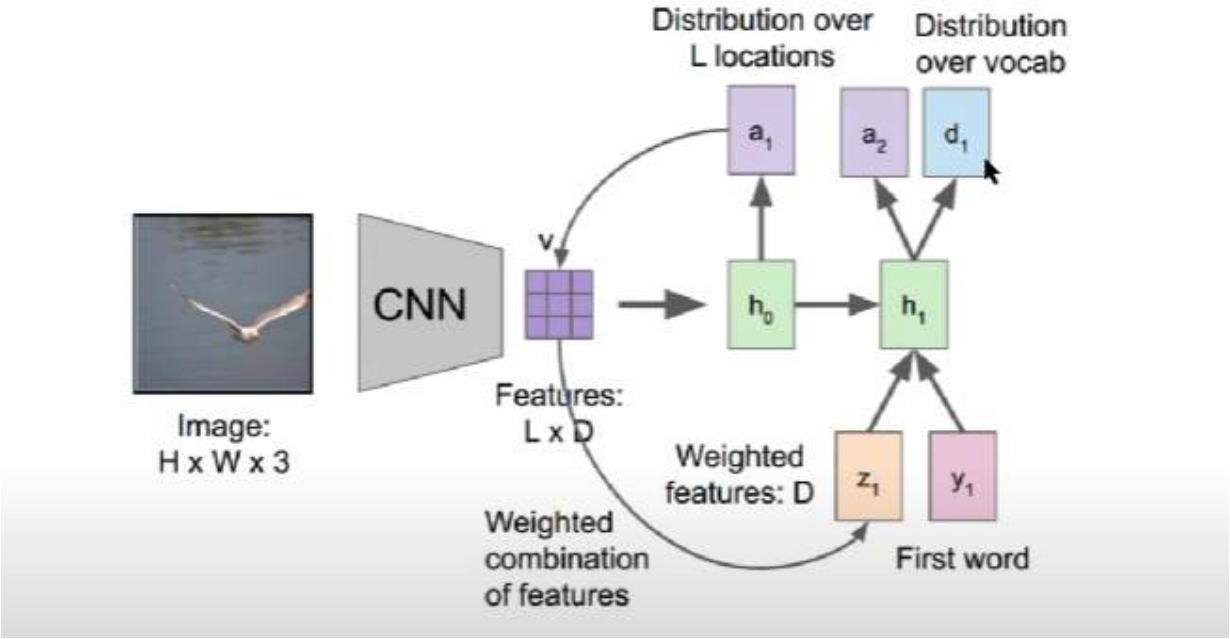
Vi=It is the features inside each of the grid locations show below



Distribution over L location means that the softmax is trying to find the probability over location of the pixels over the image.These locations are the grids in the matrix V
The rnn basically tells us the softmax probabilities over the grid positions or pixels on this grid v.This a1 has the length of H*W.This a1 is used to weight the features in the feature map.v is the weight matrix of feature map

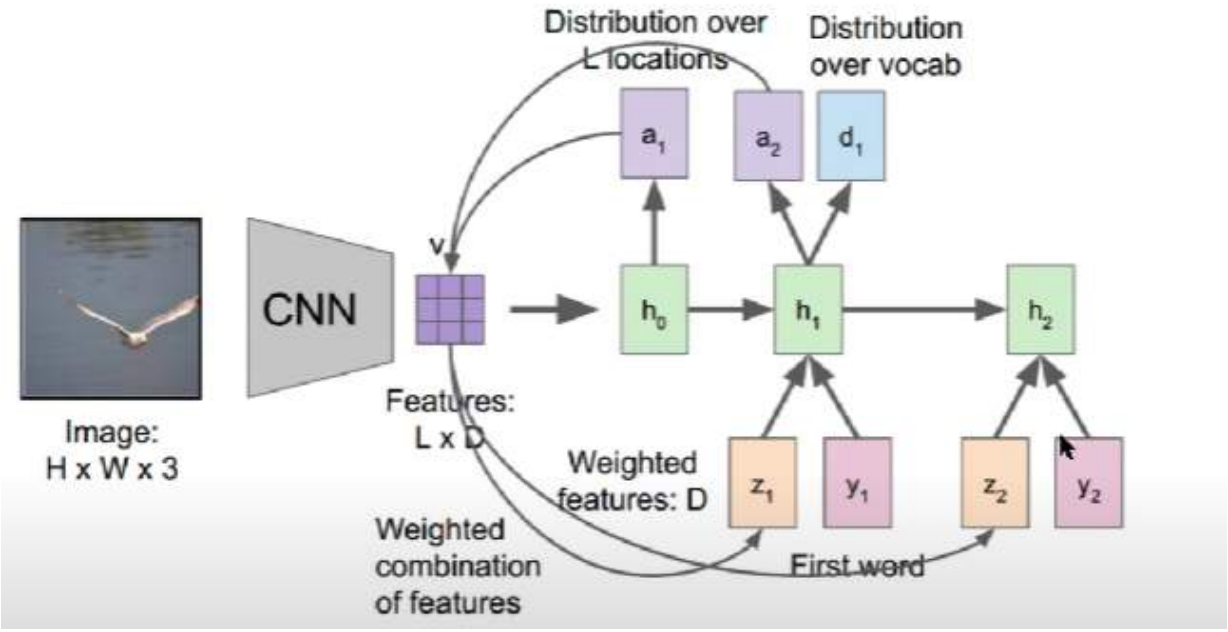


The z1 becomes input to the RNN along the first word or start token.We get the following :

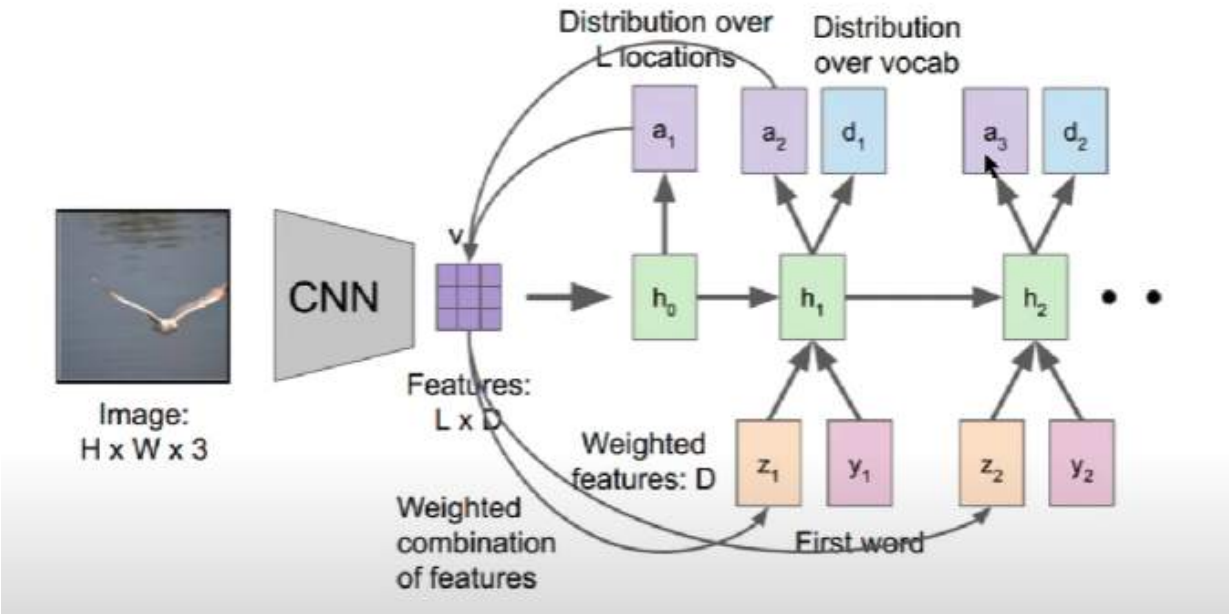


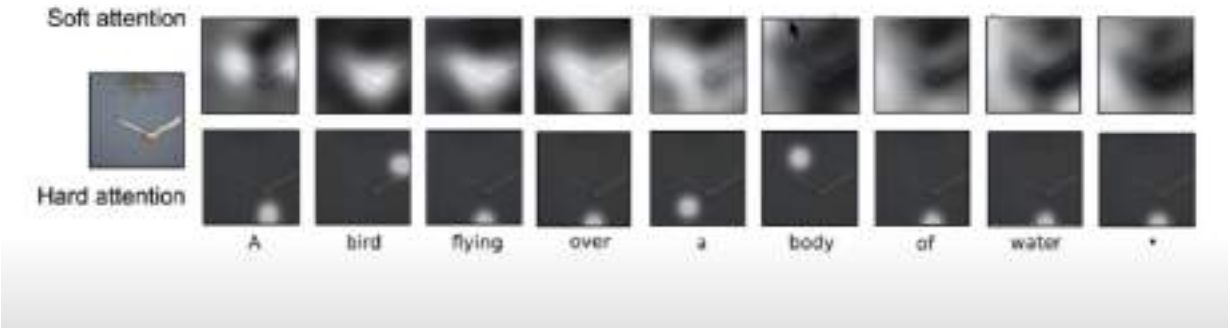
a2=distribution over l locations on the grid map

d1=distribution or probability function on the word embedding



d1 is the input as the next word to the RNN.





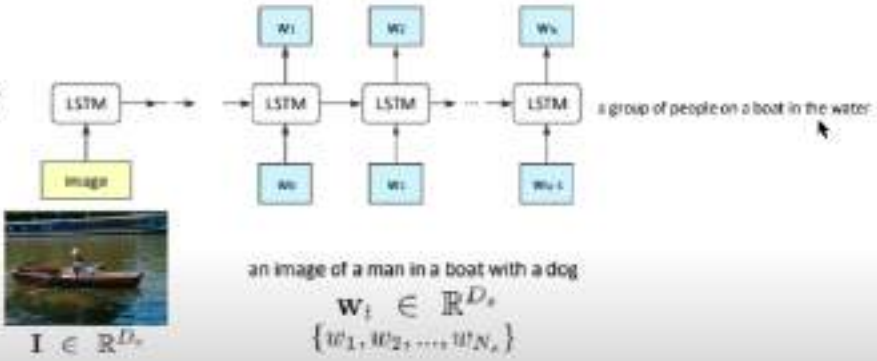
Soft Attention :The value is a softmax output which is then used to calculate zi.It is the weighted combination of all possible locations and backpropagation can be used
Hard Attention :The value a1 has lots of entries or probabilities and the one which has the highest values is then selected

For the words we use cross entropy error which can be then used to then fine tune the weights of the CNN .The total loss is the sum of cross entropy of each time step.

Image Captioning with Attention: Results

Recent Efforts: Boosting Image Captioning with Attributes in LSTMs

$$\mathbf{x}^{-1} = \mathbf{T}_v \mathbf{I}, \quad \mathbf{x}^t = \mathbf{T}_s \mathbf{w}_t,$$
$$\mathbf{h}^t = f\left(\mathbf{x}^t\right),$$
$$t \in \{0, \ldots, N_s - 1\}$$



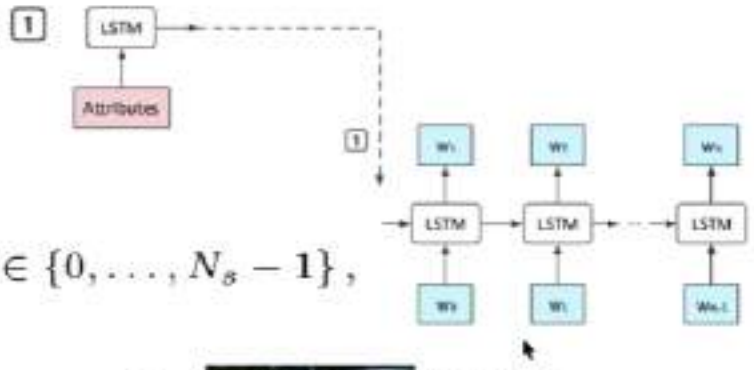
$\mathbf{I} \in \mathbb{R}^{D_v}$

$\mathbf{w}_t \in \mathbb{R}^{D_s}$

$\{w_1, w_2, \ldots, w_{N_s}\}$

Boosting Image Captioning with Attributes in LSTMs: A1

$$\mathbf{x}^{-1} = \mathbf{T}_a \mathbf{A},$$
$$\mathbf{x}^t = \mathbf{T}_s \mathbf{w}_t,$$
$$\mathbf{h}^t = f\left(\mathbf{x}^t\right), \quad t \in \{0, \ldots, N_s - 1\},$$

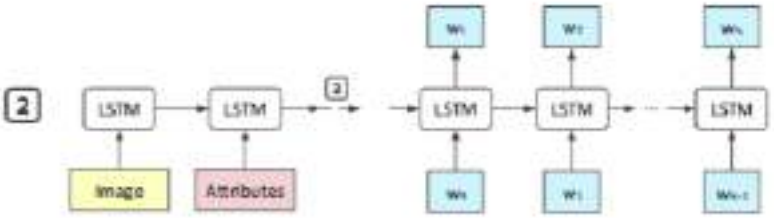


$$\mathbf{A} \in \mathbb{R}^{D_a}$$
$$\mathcal{A} = \{a_1, a_2, ..., a_{D_a}\}$$

Attributes:
boat: 1 water: 0.838 man: 0.762
riding: 0.728 dog: 0.547 small: 0.485
person: 0.471 river: 0.461

Boosting Image Captioning with Attributes in LSTMs: A2

$$\mathbf{x}^{-2} = \mathbf{T}_v \mathbf{I} \text{ and } \mathbf{x}^{-1} = \mathbf{T}_a \mathbf{A},$$
$$\mathbf{x}^t = \mathbf{T}_s \mathbf{w}_t,$$

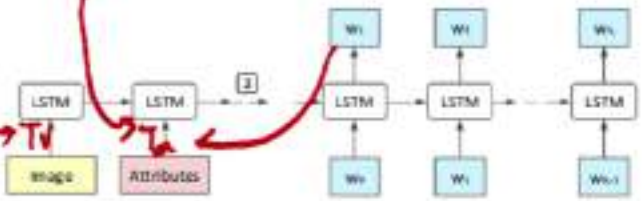


$$\mathbf{h}^t = f\left(\mathbf{x}^t\right), \quad t \in \{0, \ldots, N_s - 1\}$$

Boosting Image Captioning with Attributes in LSTMs: A2

$$\mathbf{x}^{-2} = \mathbf{T}_v \mathbf{I} \quad \text{and} \quad \mathbf{x}^{-1} = \mathbf{T}_a \mathbf{A},$$

$$\mathbf{x}^t = \mathbf{T}_s \mathbf{w}_t,$$



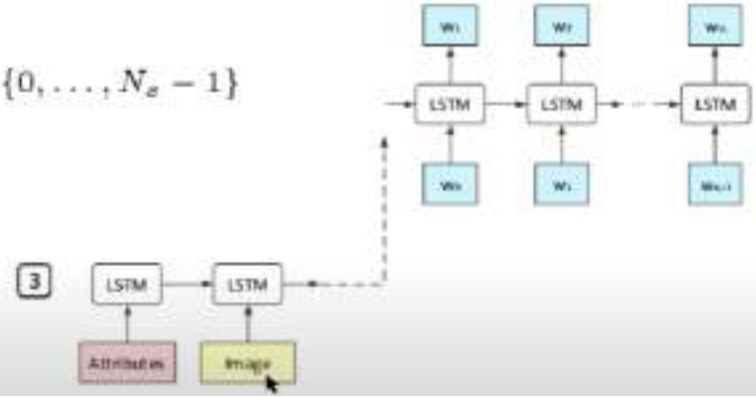
$$\mathbf{h}^t = f\left(\mathbf{x}^t\right), \quad t \in \{0, \ldots, N_s - 1\}$$

Boosting Image Captioning with Attributes in LSTMs: A3

$$\mathbf{x}^{-2} = \mathbf{T}_a \mathbf{A} \quad \text{and} \quad \mathbf{x}^{-1} = \mathbf{T}_v \mathbf{I},$$

$$\mathbf{x}^t = \mathbf{T}_s \mathbf{w}_t,$$

$$\mathbf{h}^t = f\left(\mathbf{x}^t\right), \quad t \in \{0, \ldots, N_s - 1\}$$



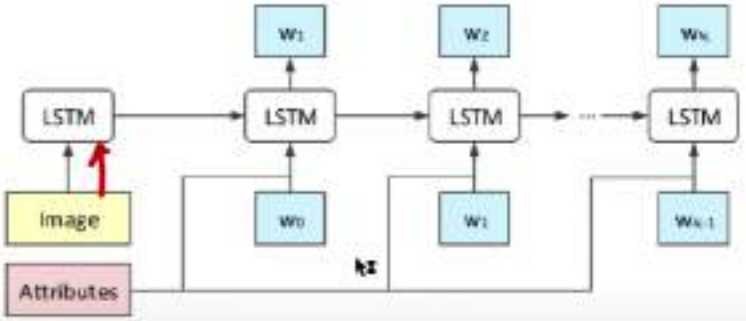
Boosting Image Captioning with Attributes in LSTMs: A5

$$\mathbf{x}^{-1} = \mathbf{T}_v \mathbf{I}, \quad (5)$$

$$\mathbf{x}^t = \mathbf{T}_s \mathbf{w}_t + \mathbf{T}_a \mathbf{A},$$

$$\mathbf{h}^t = f\left(\mathbf{x}^t\right),$$

$$t \in \{0, \ldots, N_s - 1\}$$



Boosting Image Captioning with Attributes in LSTMs: Observations

- LSTM-A_1 > LSTM
  - Indicates advantage of exploiting high-level attributes than image representations
- LSTM-A_2 > LSTM-A_1
  - Integrating image representations performs better
- LSTM-A_3 > LSTM-A_2
  - Benefits from mechanism of first feeding high-level attributes into LSTM instead of starting from image representations
- LSTM-A_4 < LSTM-A_3
  - This may be because noise in image can be explicitly accumulated, and thus network overfits more easily
  - But LSTM-A_5 which feeds attributes at each time step shows improvements on LSTM-A_3

StyleNet: Generating Attractive Visual Captions with Styles

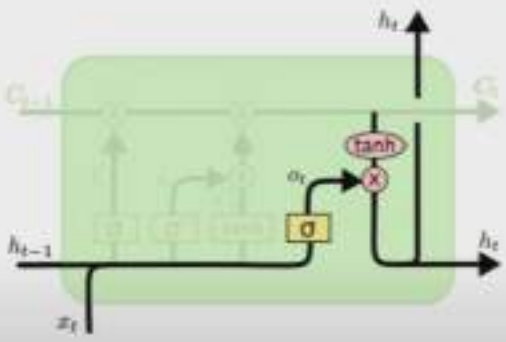
CaptionBot: A man on a rocky hillside next to a stone wall.

Romantic: A man uses rock climbing to conquer the high.

Humorous: A man is climbing the rock like a lizard.

StyleNet: Generating Attractive Visual Captions with Styles

StyleNet proposes a factored LSTM



$$W_x = U_x S_x V_x$$

$$i_t = \text{sigmoid}(W_{ix}x_t + W_{ih}h_{t-1})$$

$$f_t = \text{sigmoid}(W_{fx}x_t + W_{fh}h_{t-1})$$

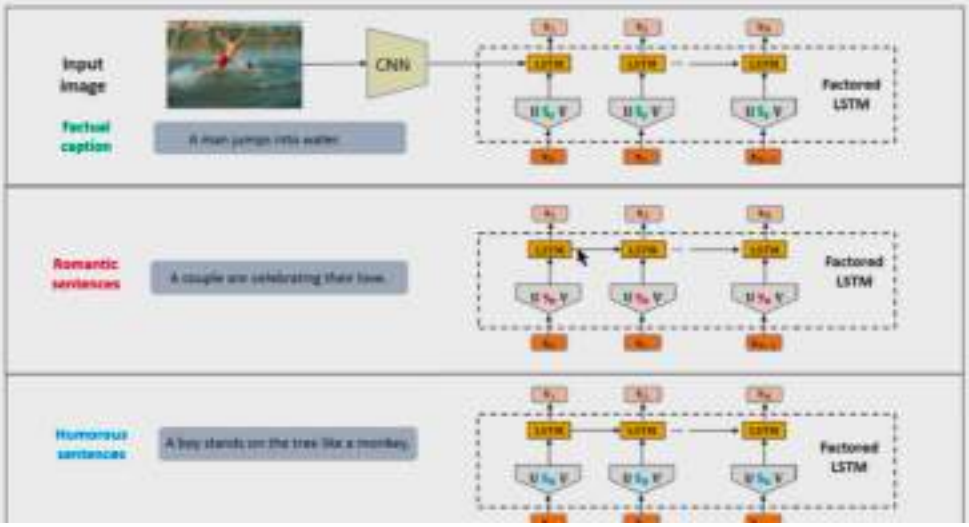
$$o_t = \text{sigmoid}(W_{ox}x_t + W_{oh}h_{t-1})$$

$$\tilde{c}_t = \tanh(W_{cx}x_t + W_{ch}h_{t-1})$$

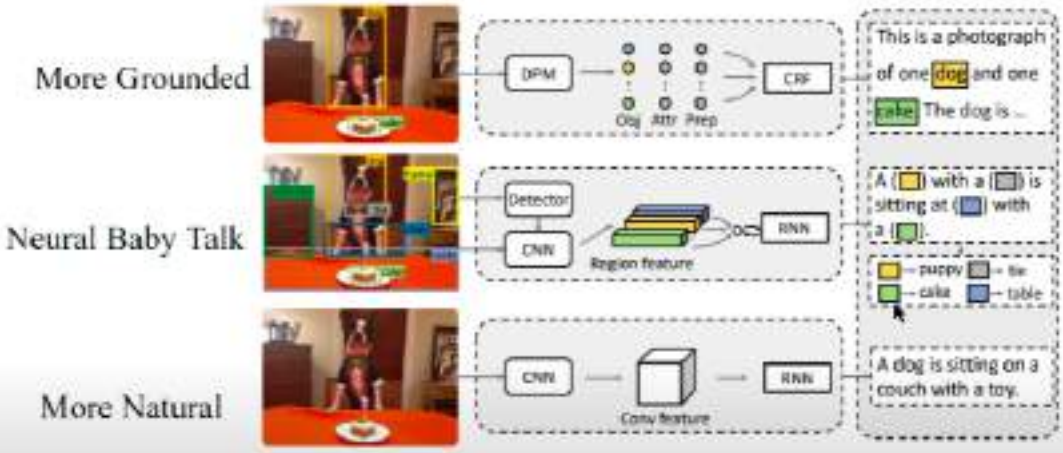
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

$$h_t = o_t \odot c_t$$

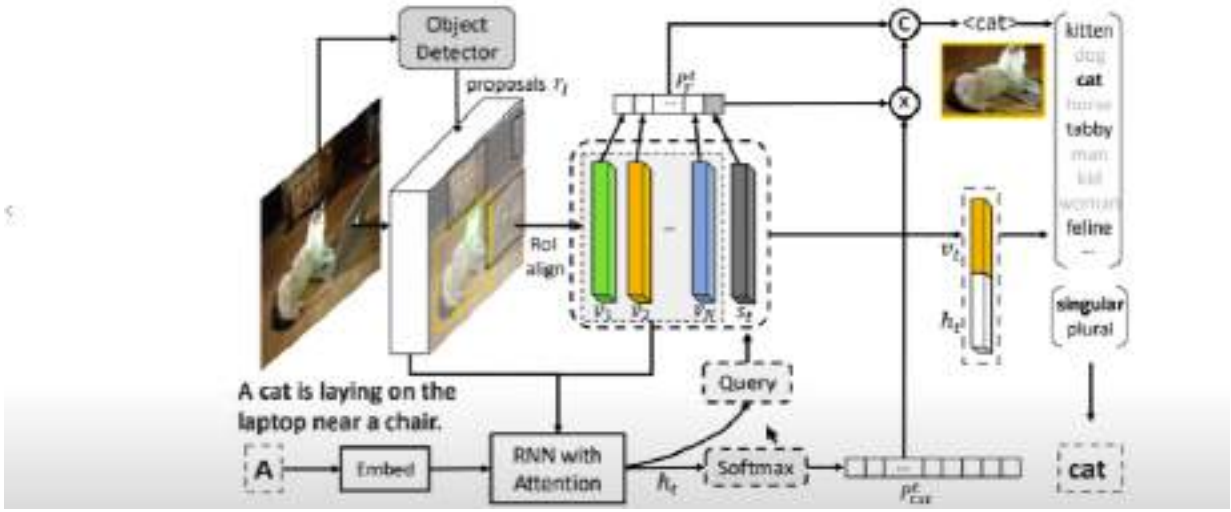
StyleNet: Generating Attractive Visual Captions with Styles



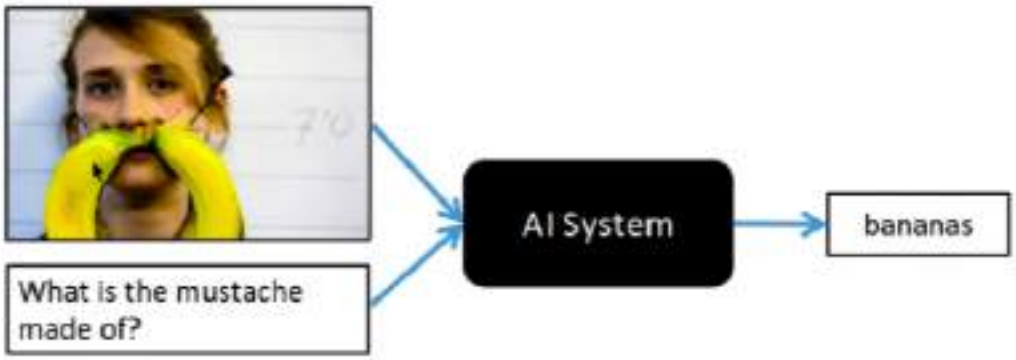
Neural Baby Talk



Neural Baby Talk



Visual Question Answering (VQA): Task Overview[1]



VQA Dataset[2]

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?

VQA Dataset[2]

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?

- **Open-ended answers** and **Multiple-choice answers**
- $250K$ images (MS COCO + $50K$ abstract images)

VQA Dataset[2]

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?

- **Open-ended answers** and **Multiple-choice answers**
- $250K$ images (MS COCO + $50K$ abstract images)
- $750K$ questions, $10M$ answers
- Each question is answered by 10 human annotators

COCO-QA[3]

COCOQA 5078
How many leftover donuts is the red bicycle holding?
Ground truth: three

COCOQA 1238
What is the color of the tee-shirt?
Ground truth: blue

COCOQA 26088
Where is the gray cat sitting?
Ground truth: window

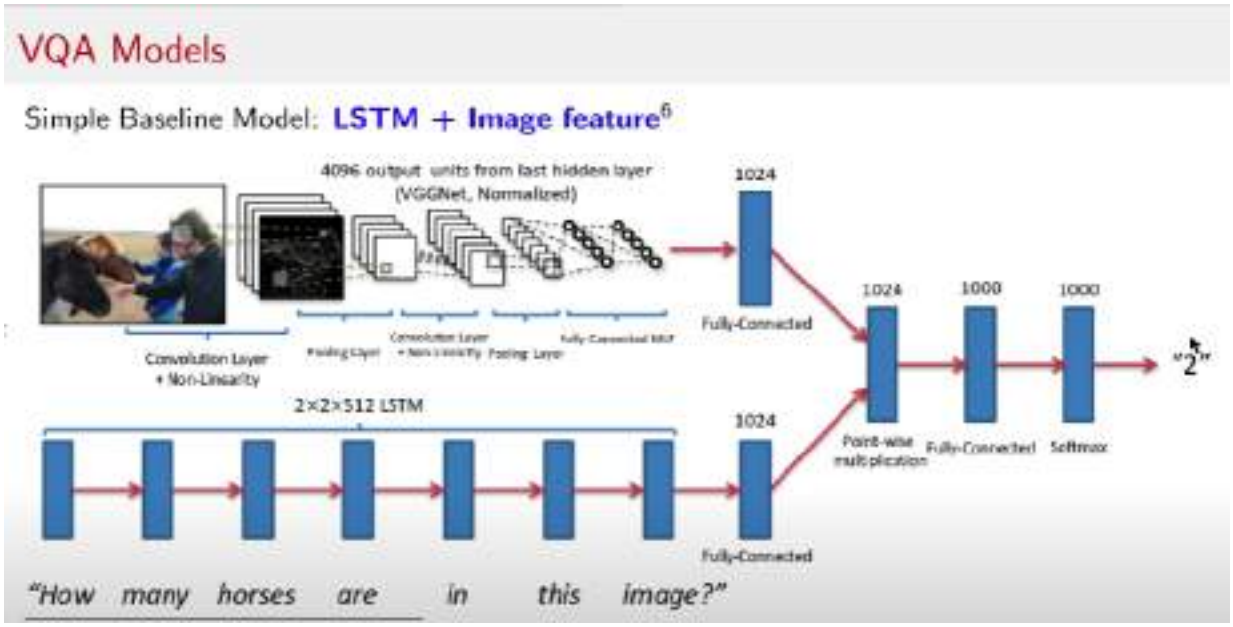
- Automatically generate QA pairs with MS COCO captions
- $118K$ QA pairs on $123K$ images
- 4 types of questions: **What object, How many, What color, Where**

Visual7W[4]

- 7W stands for what, where, when, who, why, how and which
- $328K$ QA pairs on $\sim 47K$ images
- Two types of tasks: **telling** and **pointing**

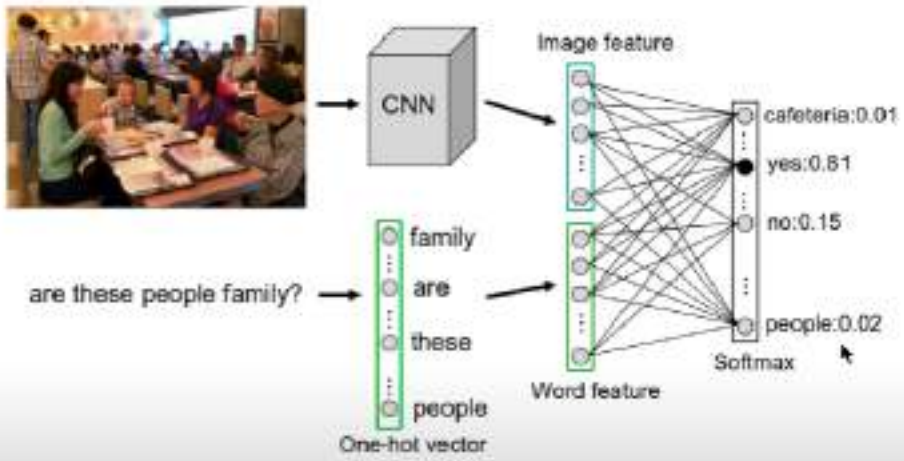
CLEVR[5]

- $100K$ **rendered images** and $1M$ **automatically generated questions**
- Questions are complex and require reasoning skills

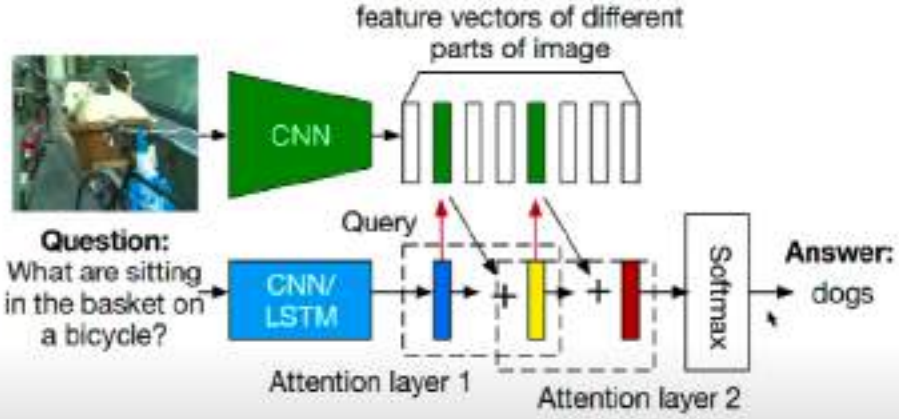


In the above we can use backprop and gradient descent easily

VQA Models: Bag-of-words + Image Features (iBOWIMG)[7]



VQA Models: Stacked Attention Networks for Image Question Answering[8]



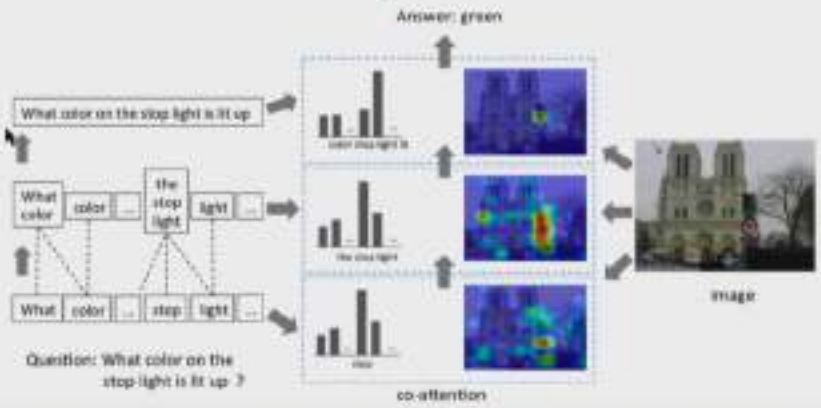
VQA Models: Stacked Attention Networks for Image Question Answering[8]

Original Image First Attention Layer Second Attention Layer

The stacked attention network **first focuses on all referred concepts**, e.g., *bicycle*, *basket* and objects in the basket (*dogs*) in the first attention layer and **then further narrows down the focus in the second layer** and finds out the answer "**dog**".

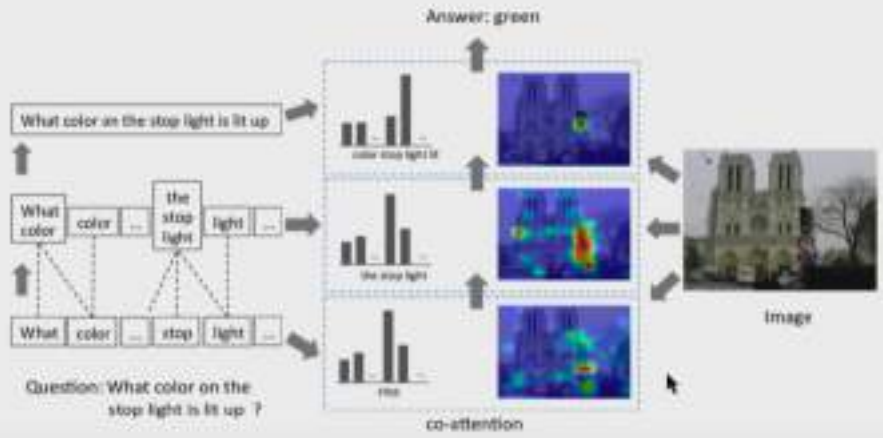
Can we do attention on question as well?

VQA Models: Hierarchical Co-Attention Model[9]



Given a question, extract its word level, phrase level and question level embeddings

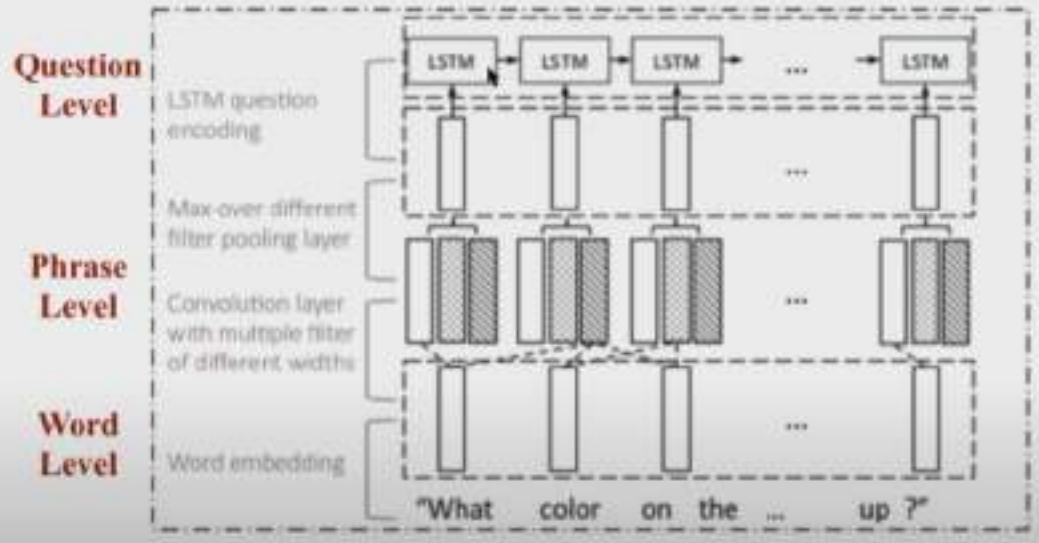
VQA Models: Hierarchical Co-Attention Model[9]



Given a question, extract its word level, phrase level and question level embeddings
At each level, apply co-attention on both image and question
Final answer prediction is based on all co-attended image and question features

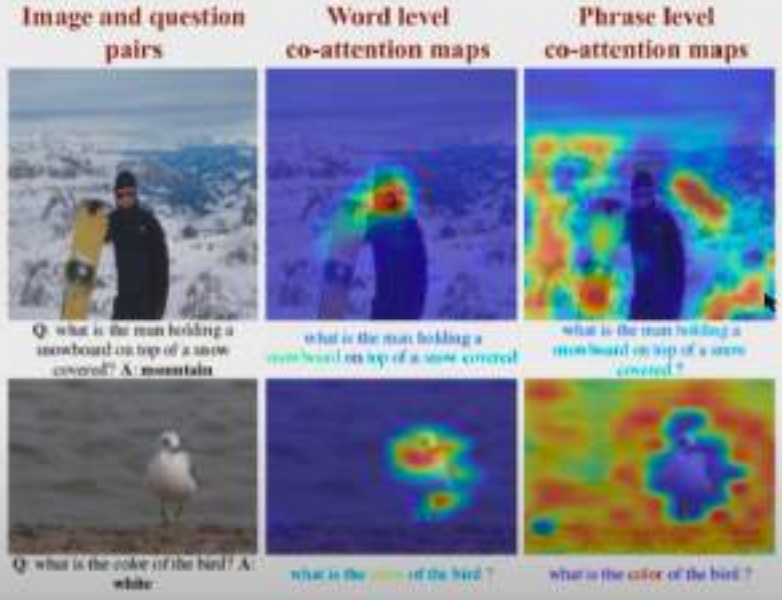
VQA Models: Hierarchical Co-Attention Model

Question hierarchy



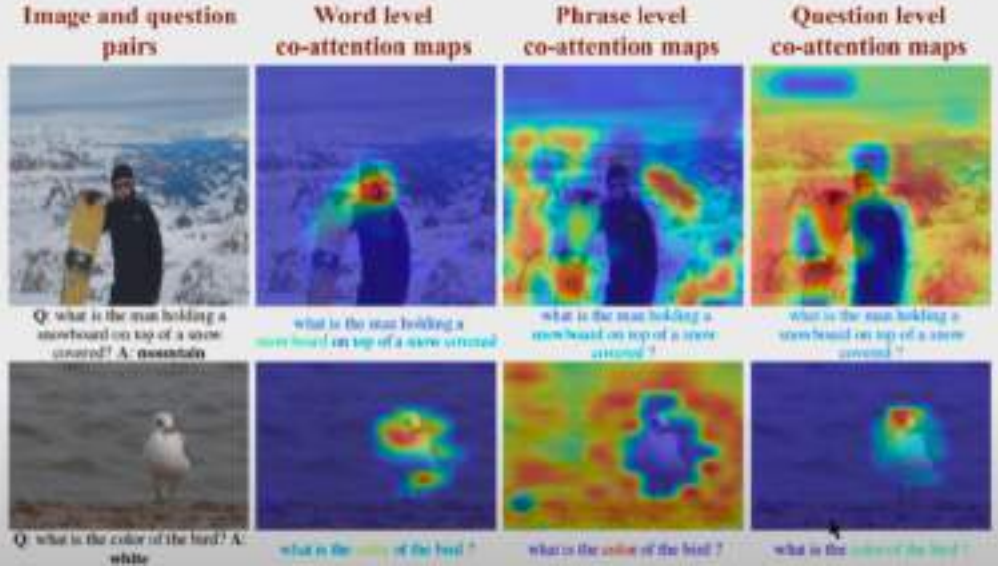
VQA Models: Hierarchical Co-Attention Model

Visualization of image and question co-attention map



VQA Models: Hierarchical Co-Attention Model

Visualization of image and question co-attention map



Visual Dialog: Task Description

- **Given**
  - Image I
  - Human dialog history $(Q_1, A_1), (Q_2, A_2), ..., (Q_{t-1}, At-1)$
  - Follow-up question Q_t
- **Task**
  - Produce free-form natural language answer A_t

Q: How many people on wheelchairs?
A: Two
Q: What gender are the people in the wheelchairs?
A: One is female, one is male
Q: Which one is holding the racket?
A: The female

Q: Is the other one holding anything?

A: He is not

Visual Dialog: Evaluation

- A fundamental challenge in dialog systems is **evaluation**[11]
- Existing **word-overlap based metrics** such as BLEU, METEOR, ROUGE are known to correlate poorly with human judgement

Visual Dialog: Evaluation Protocol

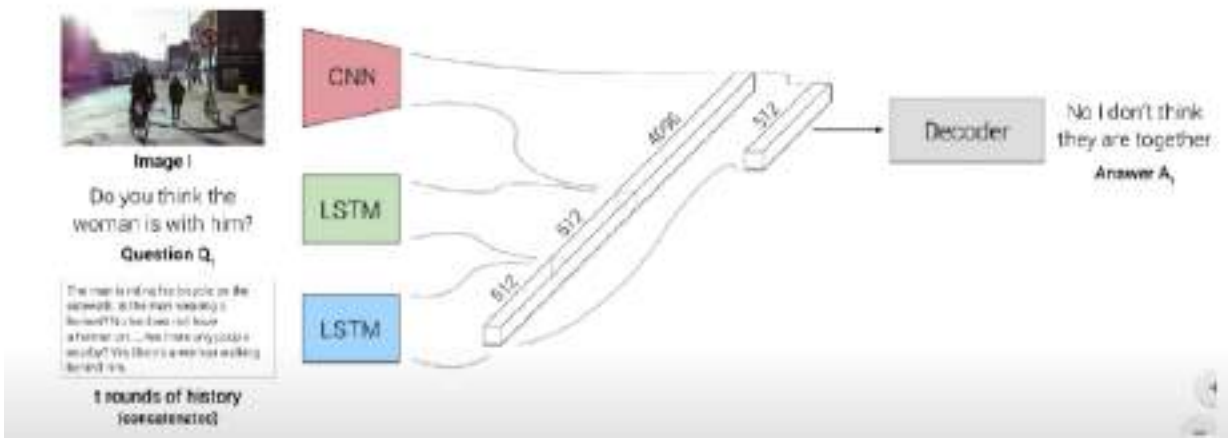
- **Given**
  - Image I
  - Human dialog history $(Q_1, A_1), (Q_2, A_2), ..., (Q_{t-1}, At-1)$
  - Follow-up question Q_t
  - 100 answer options
    - Answers to 50 most similar questions
    - 30 popular answers
    - 20 random answers
- **Evaluation Task**
  - Rank the list of 100 options
- **Accuracy/Error**
  - Mean rank w.r.t. ground truth, Mean reciprocal rank

Q: How many people on wheelchairs?
A: Two
Q: What gender are the people in the wheelchairs?
A: One is female, one is male
Q: Which one is holding the racket?
A: The female

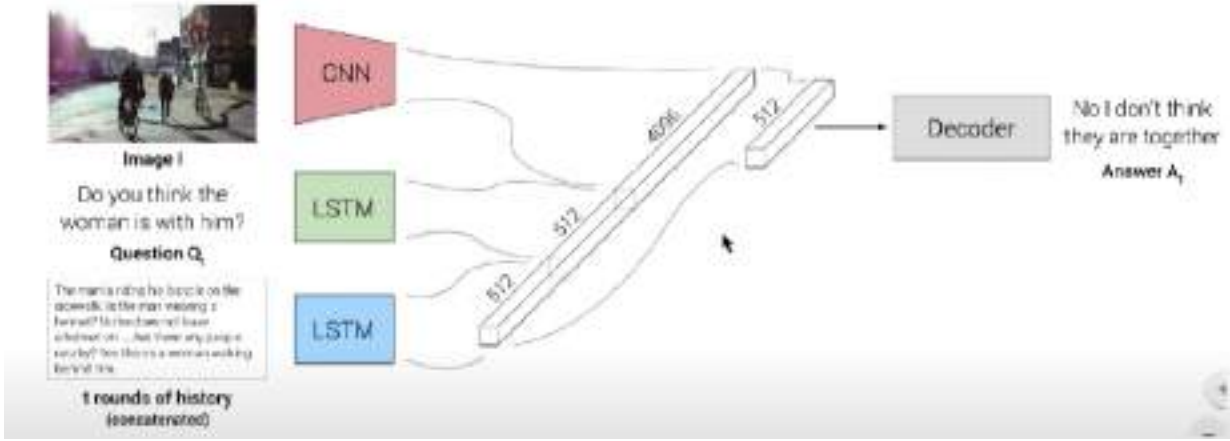
Q: Is the other one holding anything?

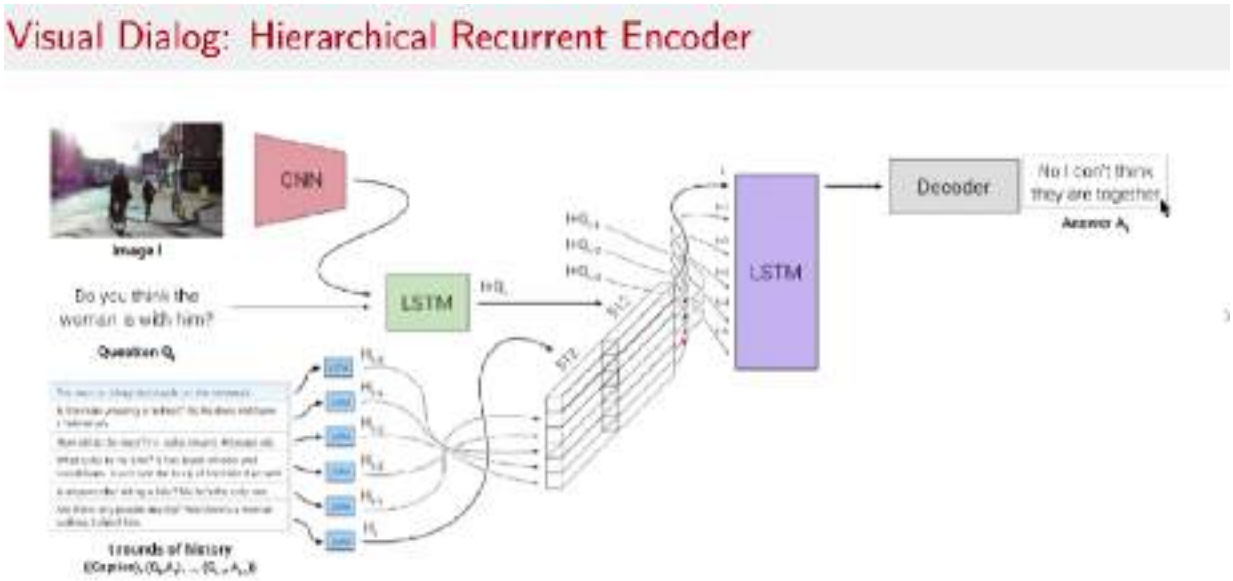
A: He is not

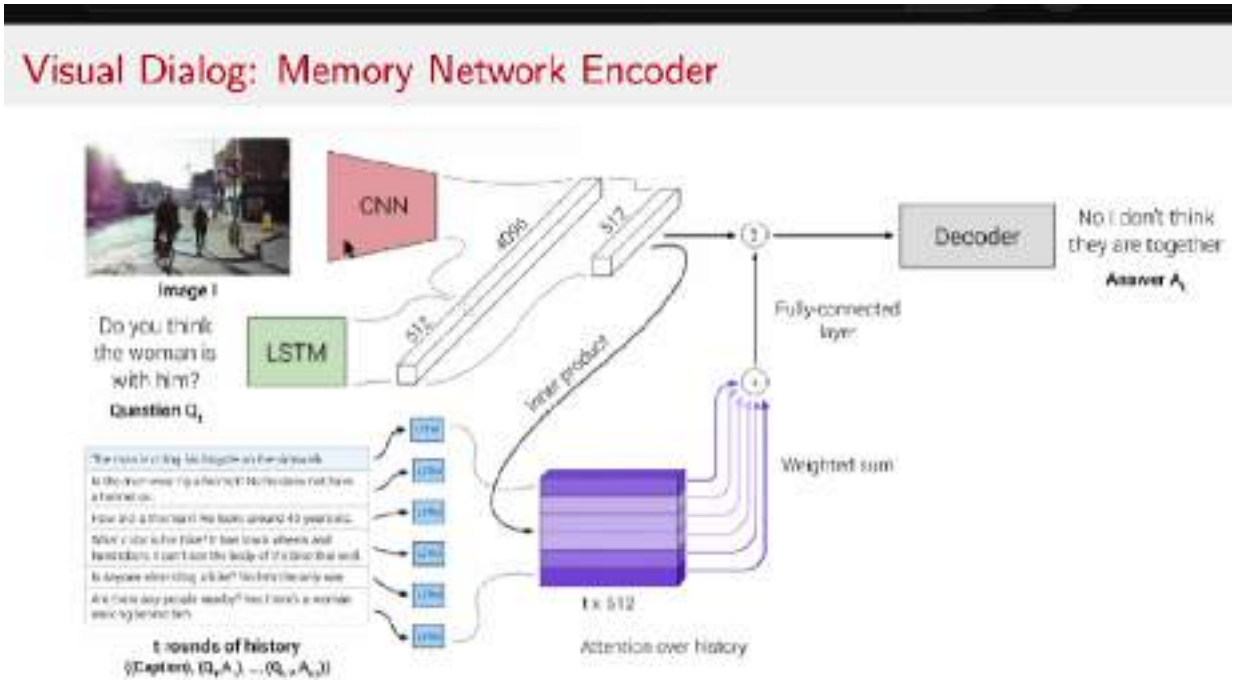
Visual Dialog: Late Fusion Encoder



Visual Dialog: Late Fusion Encoder





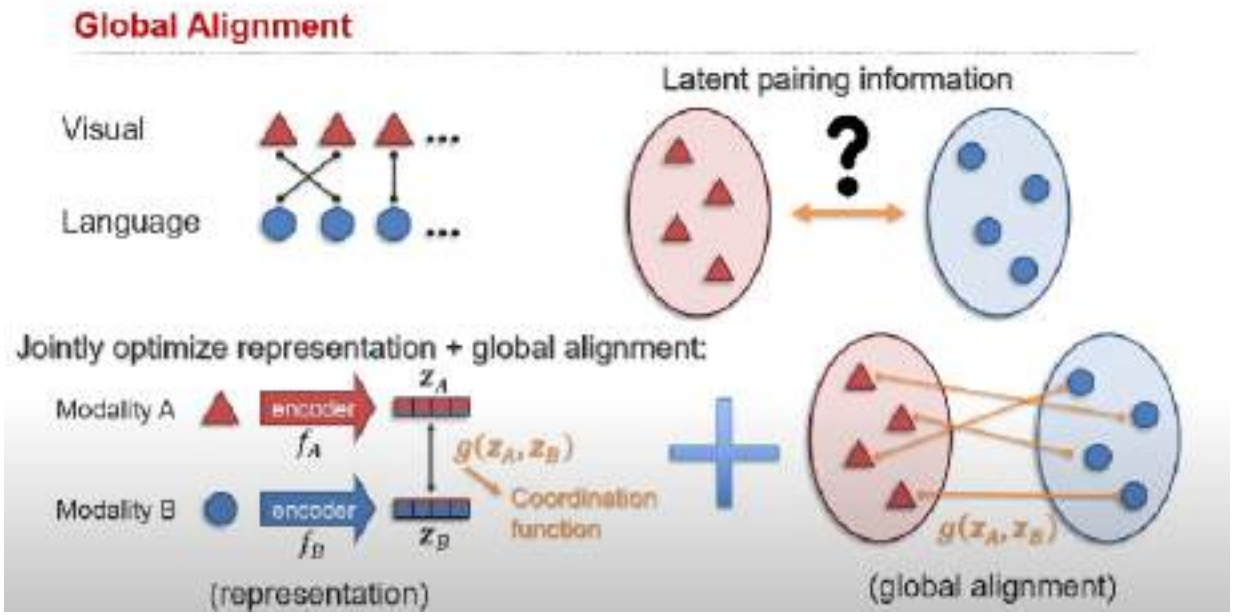


Back to Carnage Mellon University Videos:

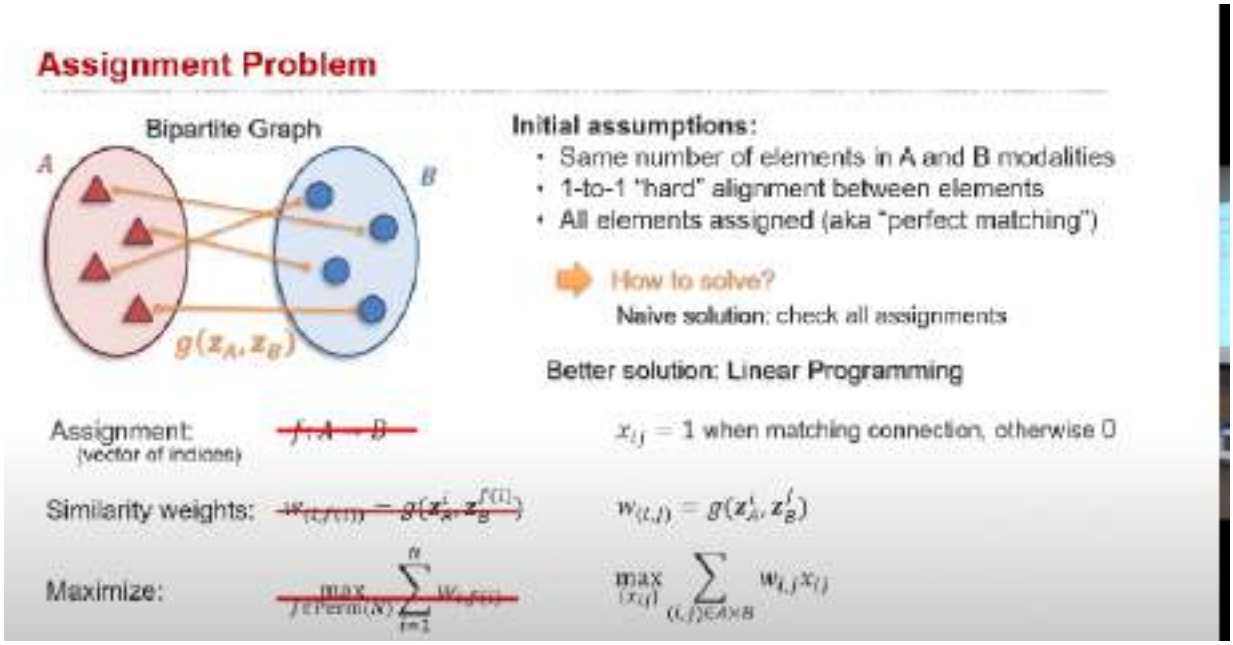
Soft Attention :Computing the gradient which will be much easier as it consists of softmax functions
Hard Attention

Image Caption is the case of direct Alignment
Hard Attention:

Global Alignment:

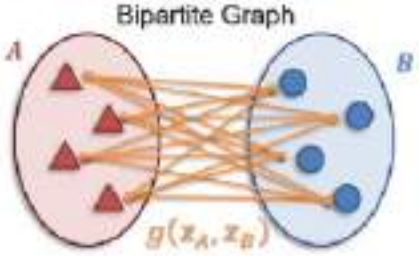


The problem of global alignment is assignment problem:



The problem is that we have to do many permutations of matching elements from one element in modality A and modality B ,this is done with the help of similarity weights but it needs to be the maximum which is computationally expensive

Optimal transport



New assumptions:

- Different number of elements in A and B modalities
- Many-to-many "soft" alignment between elements

➡ It can be seen as "transporting" elements from modality A to modality B (and vice-versa)

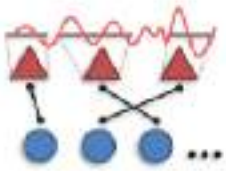
Assignments: $x_{(i,j)}$: soft alignment between z_A^i and z_B^j

Similarity weights: $w_{(i,j)} = g(z_A^i, z_B^j)$

Maximize: $\max_{\{x_{ij}\}} \sum_{(i,j)\in A\times B} w_{i,j} x_{ij}$

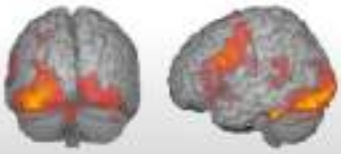
➡ Wasserstein distance give optimal transport

Discretization (aka Segmentation)



Common assumptions: ① Segmented elements

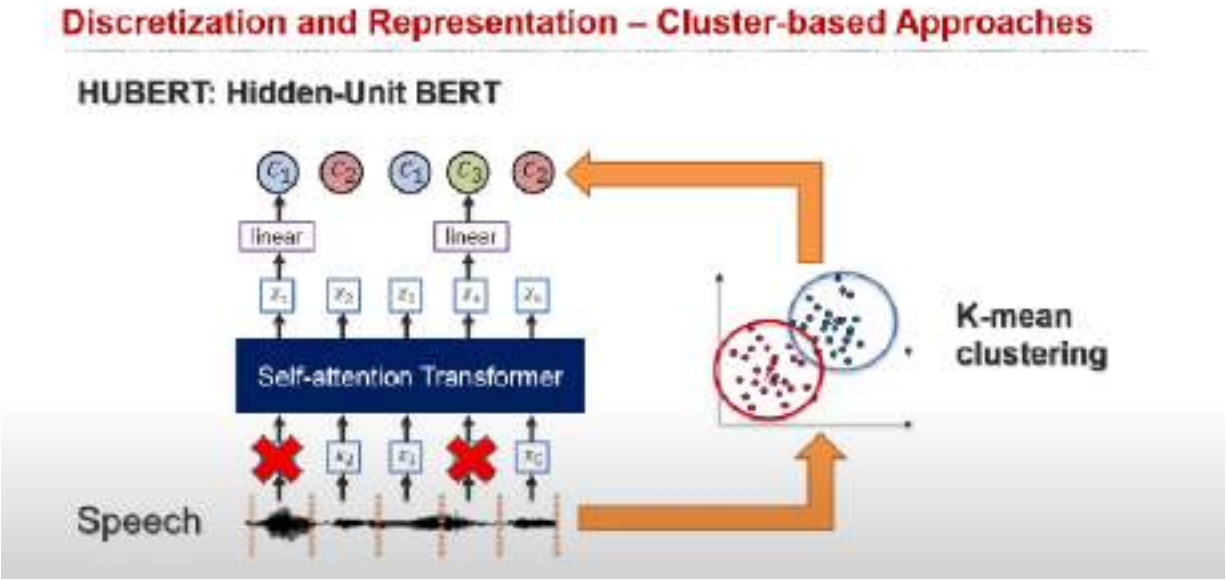
Examples:



Medical imaging

Signals

Images



In hidden bert we take the input(speech in this case) use k means to cluster it , get clustered representation and then we use self attention transformers to do prediction on the segments of the input and see which cluster does it belong to.

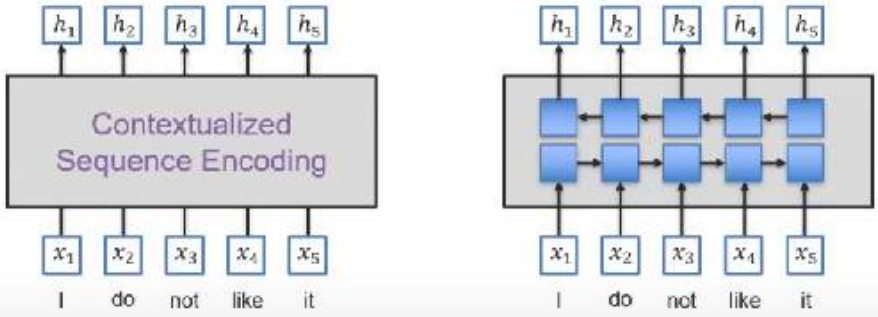
Lecture 5.2:

Aligned Representations

LSTM are not good in parallelisation i.e same time computing of different components

Sequence Encoding - Contextualization

Option 1: Bi-directional LSTM:
(e.g., ELMO)

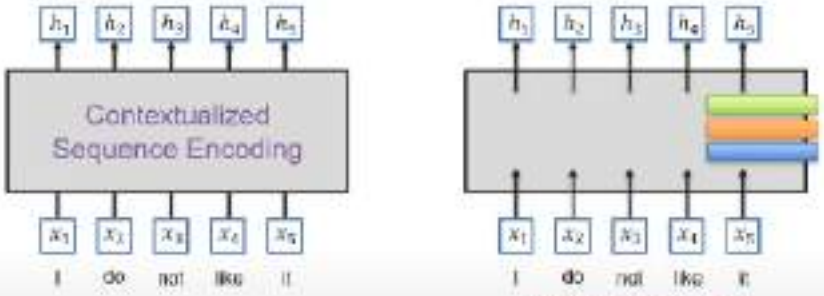


How to encode this sequence while modeling the interaction between elements (e.g., words)?

But harder to parallelize…

Sequence Encoding - Contextualization

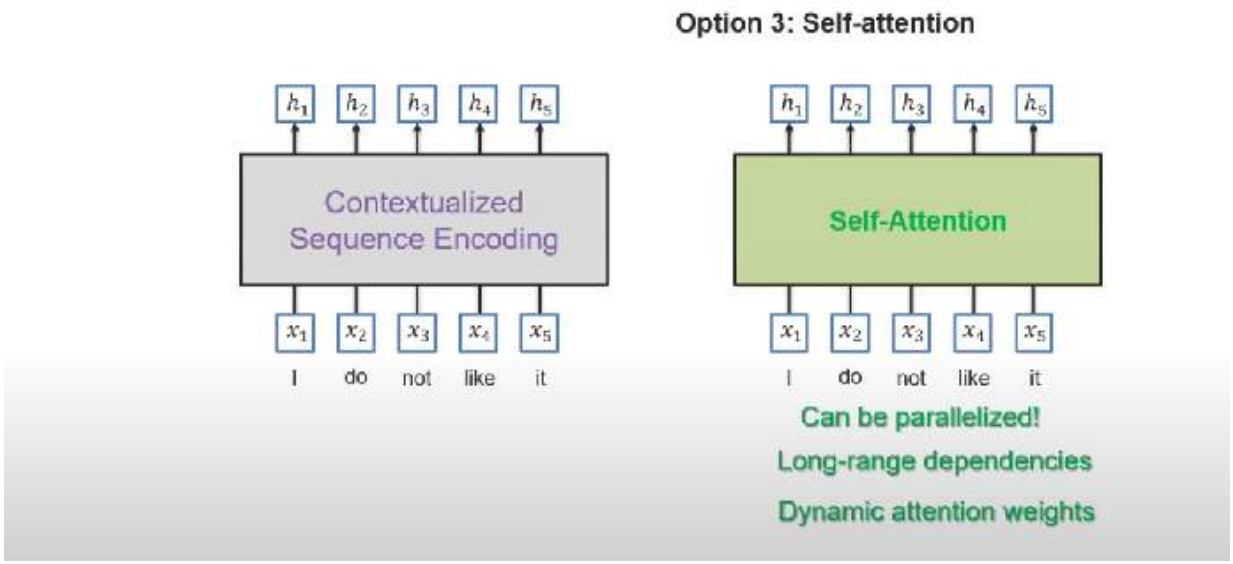
Option 2: Convolutions



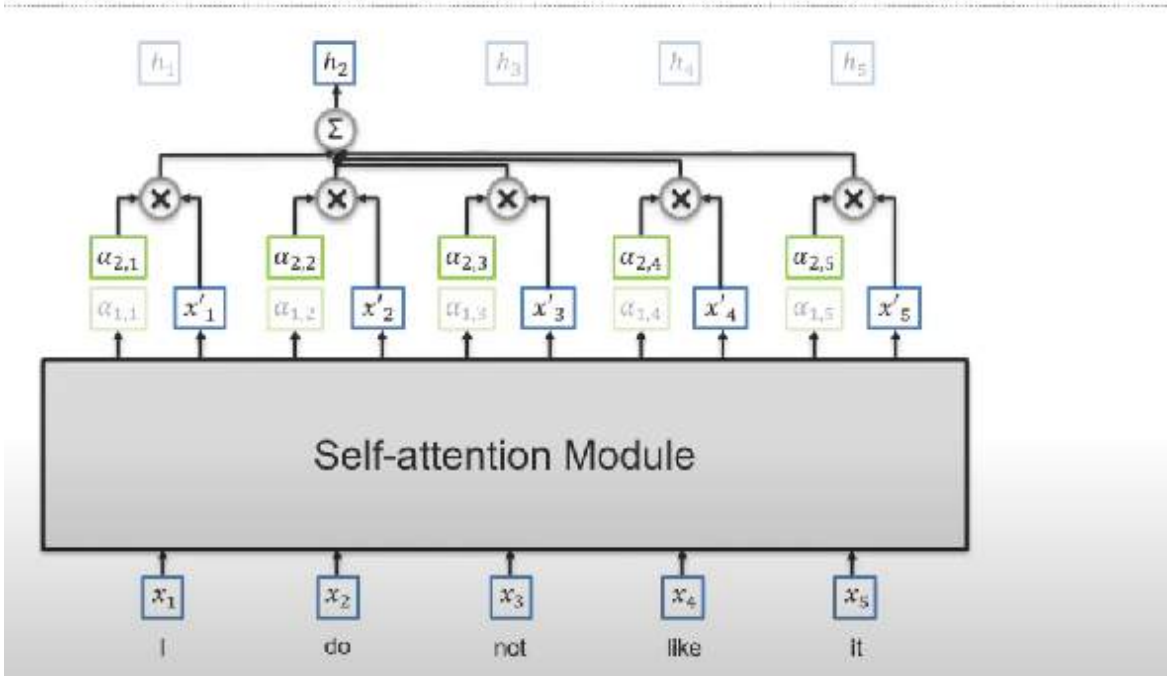
Can be parallelized!

But modeling long-range dependencies require multiple layers

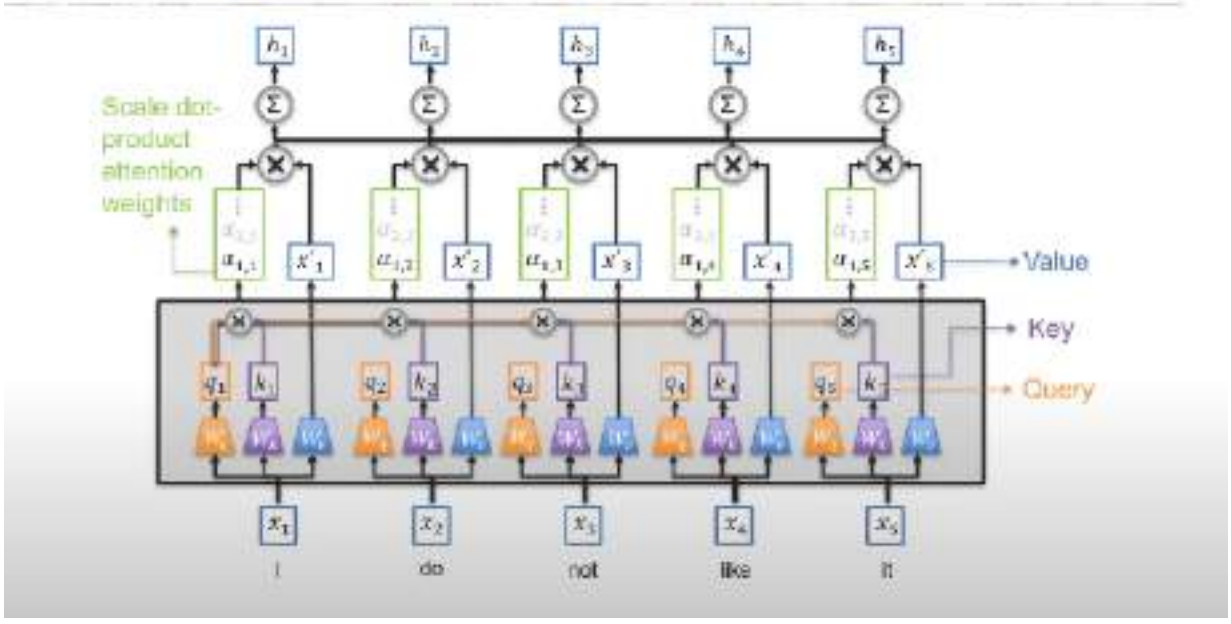
And convolutional kernels are static



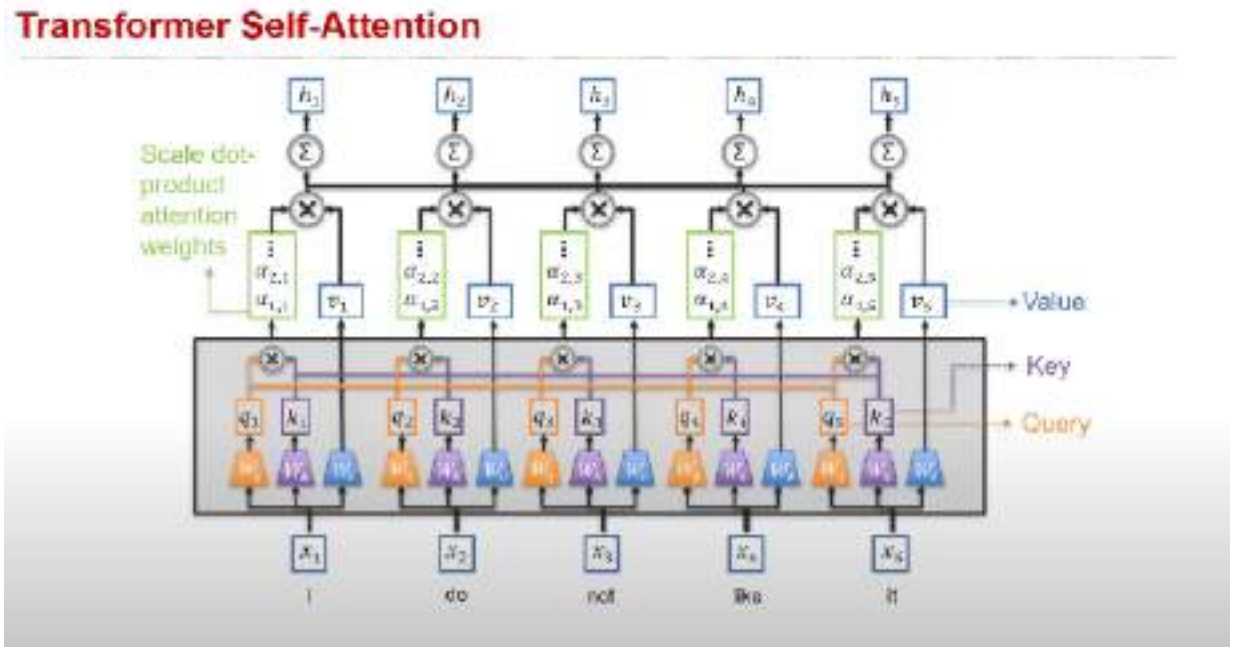
What happens in self attention is that we check what is the similarity between two word by keeping one word fixed and check how much is the similarity between them as shown below :

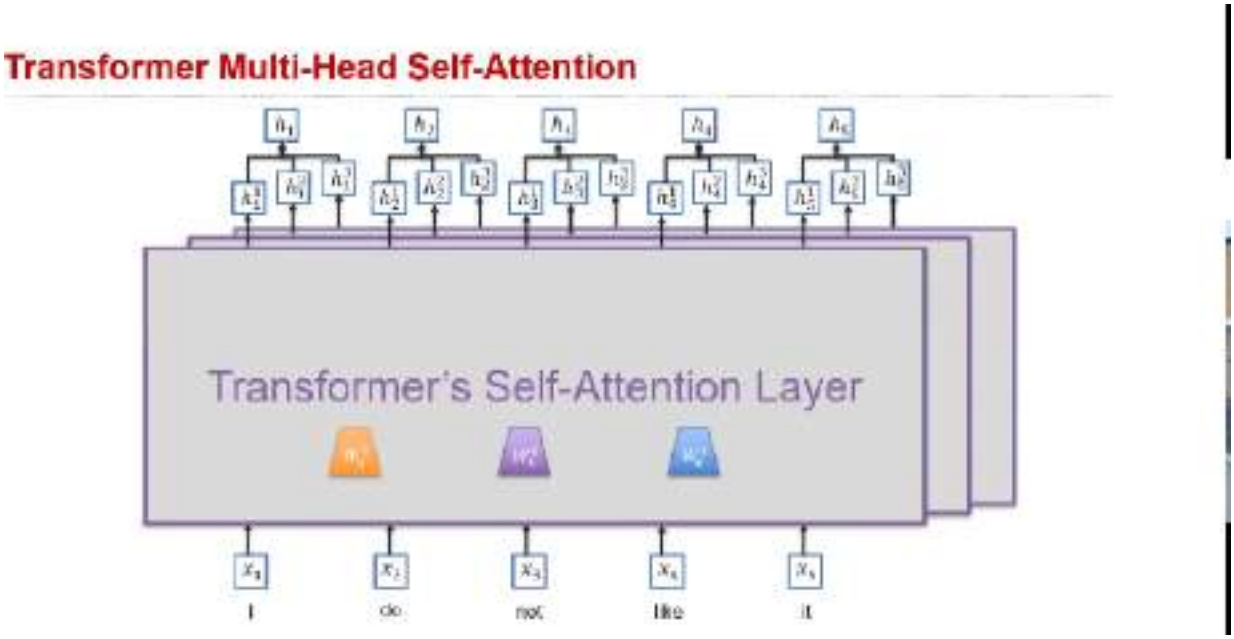


ansformer Self-Attention



The no of parameters is going to be same in comparison to rnn,alpha is usually between 0 and 1





The problem of kernels in CNN is that they are not dynamic and might not be able to do good prediction with text and are computationally expensive

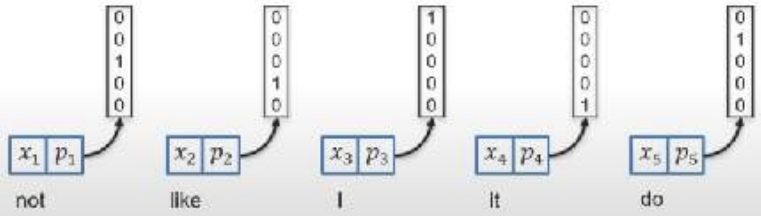
Position Embedding: 40 mins
In self attention module the postional embedding is not there ,therefore we need add it separately.

Position embeddings

- Position information is not encoded in a self-attention module

How can we encode position information?

Simple approach: one-hot encoding



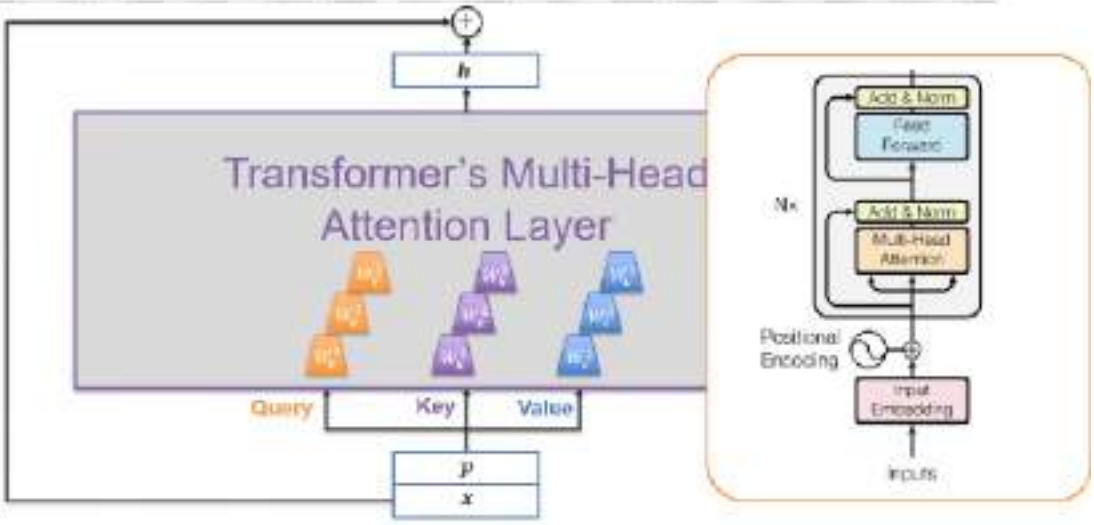
Position embeddings

- Position information is not encoded in a self-attention module

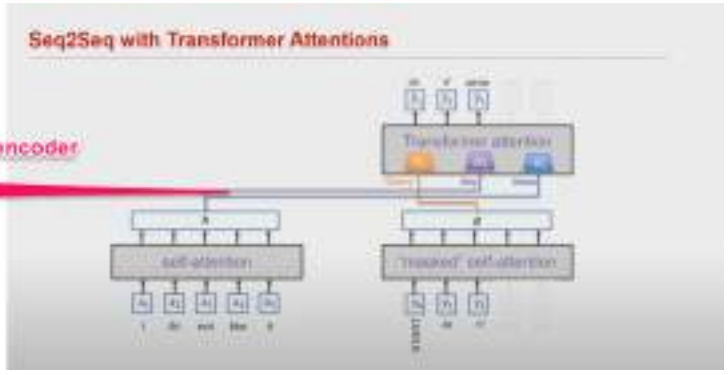
How can we encode position information?

Simple approach: one-hot encoding + linear embeddings + { Sum / - or - / concat

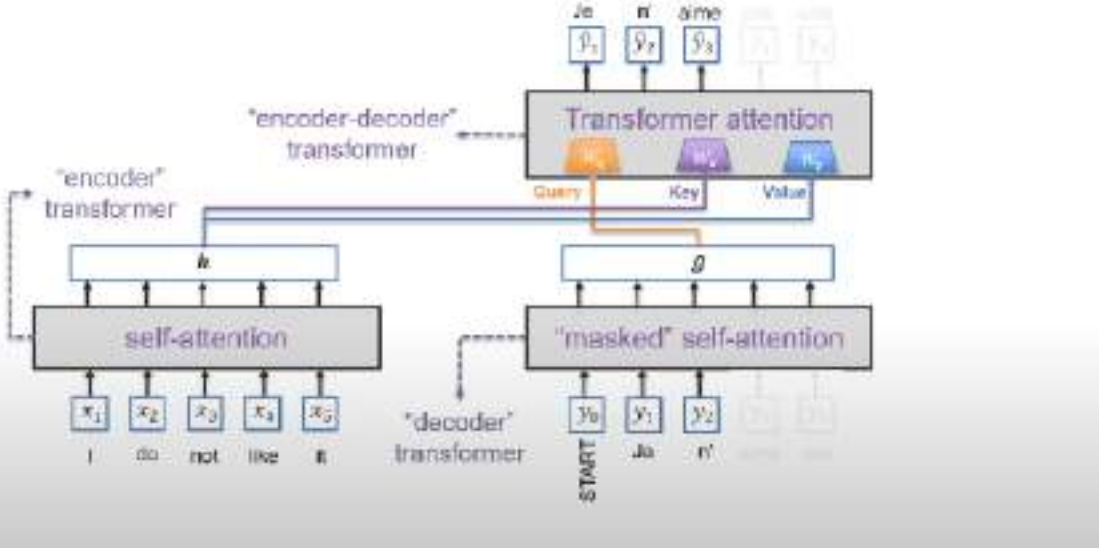
Transformer – Residual Connection



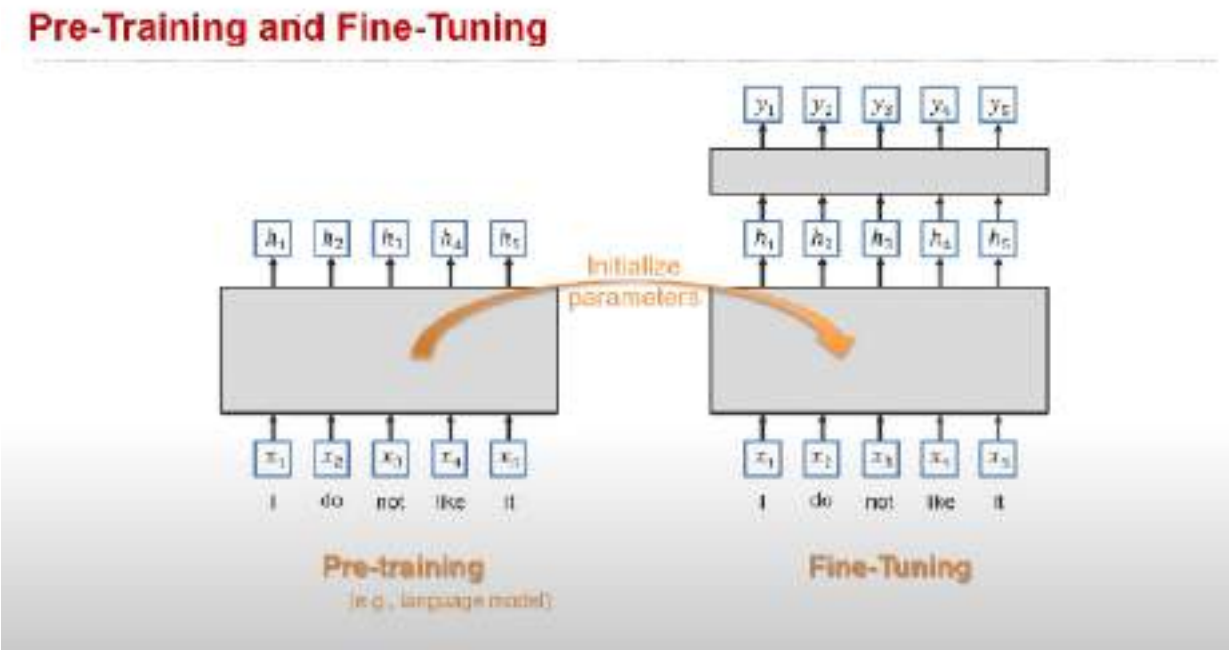
The keys and the values are coming from the encoder



Seq2Seq with Transformer Attentions



Language Pre-Training:

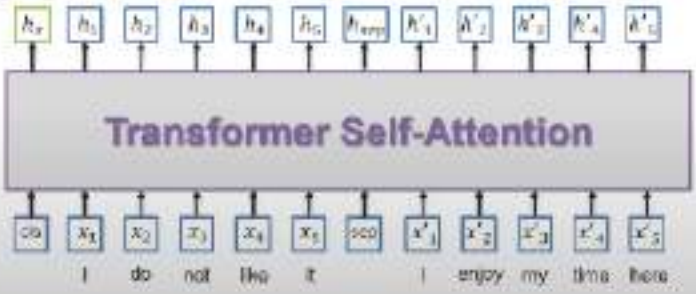


Bert learns the token level representation but also learns representational meaning of the sentence

BERT: Bidirectional Encoder Representations from Transformers

Advantages:

1. Jointly learn representation for token-level and sentence level
2. Same network architecture for pre-training and fine-tuning
3. Can be used learn relationship between sentences
4. Models bidirectional interactions between tokens

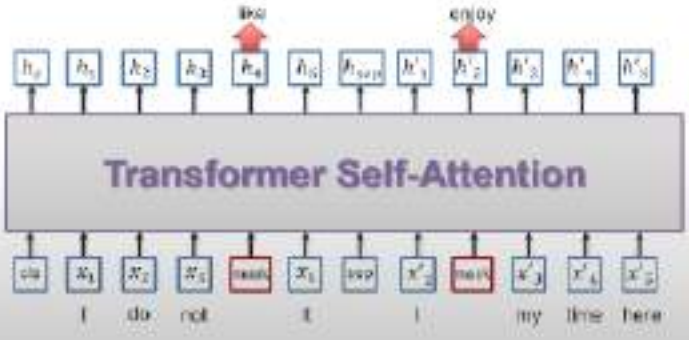


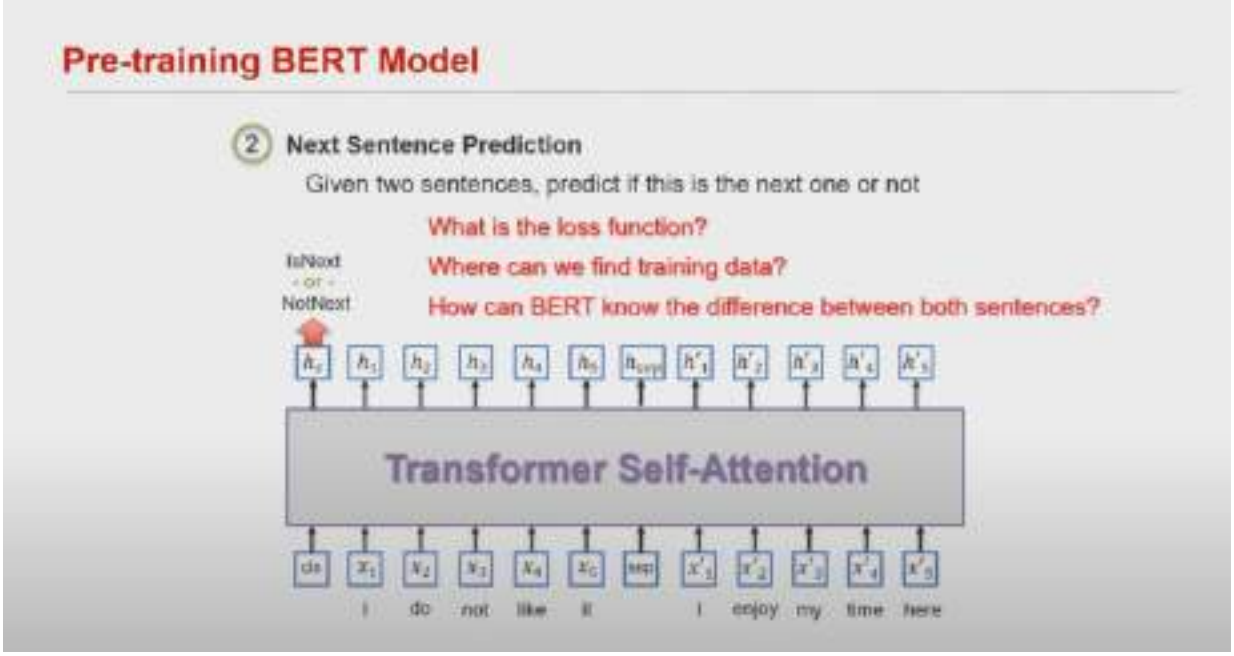
Pre-training BERT Model

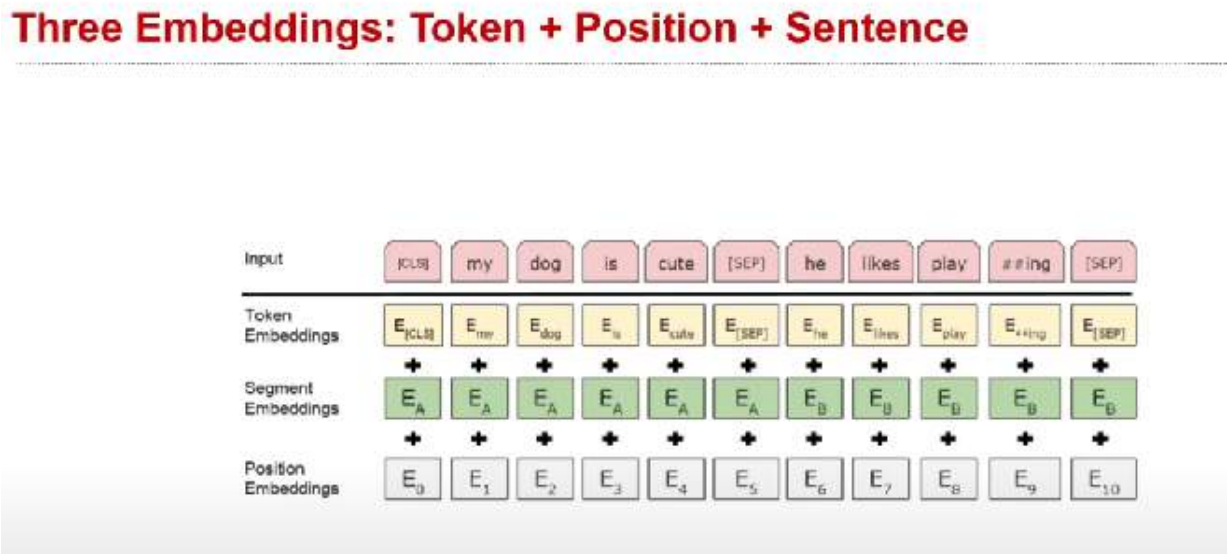
1. **Masked Language Model**

   Randomly mask input tokens and then try to predict them

   What is the loss function?







Lecture 6.2

Multimodal Aligned Representations

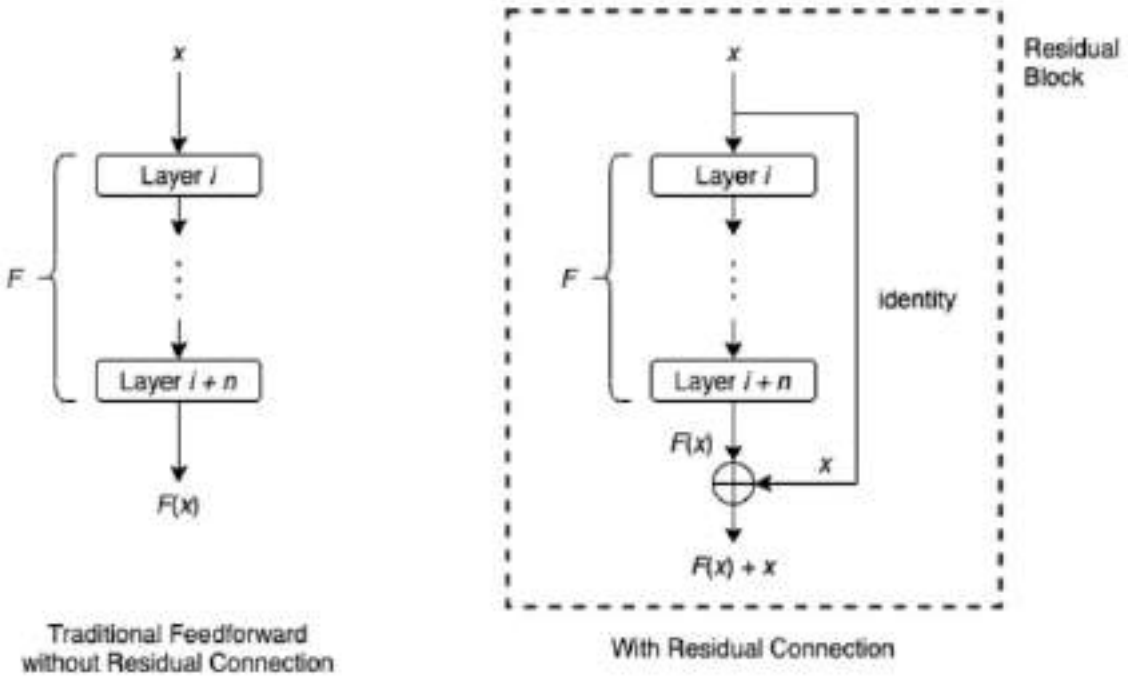
What is the point of residual connection?

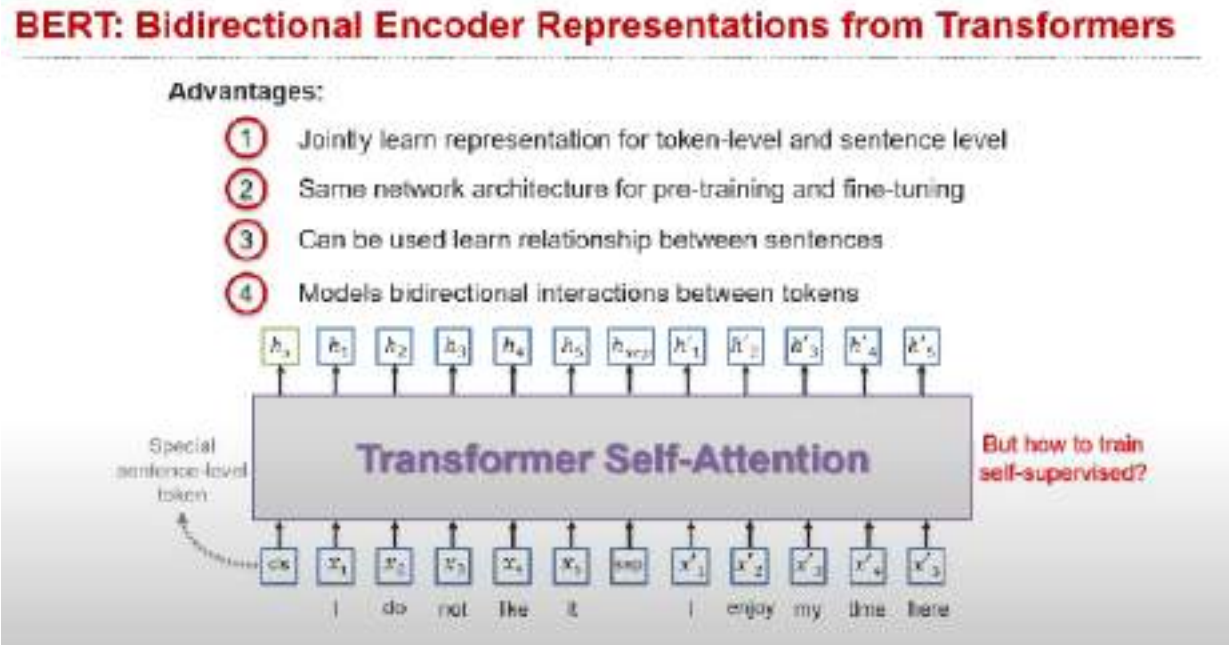
A residual connection connects the output of one earlier convolutional layer to the input of another future convolutional layer several layers later (e.g. a numer of intermediate

convolutional steps are skipped). The input of the mainstream model and the input of the convolutional prior are combined using a simple sum.

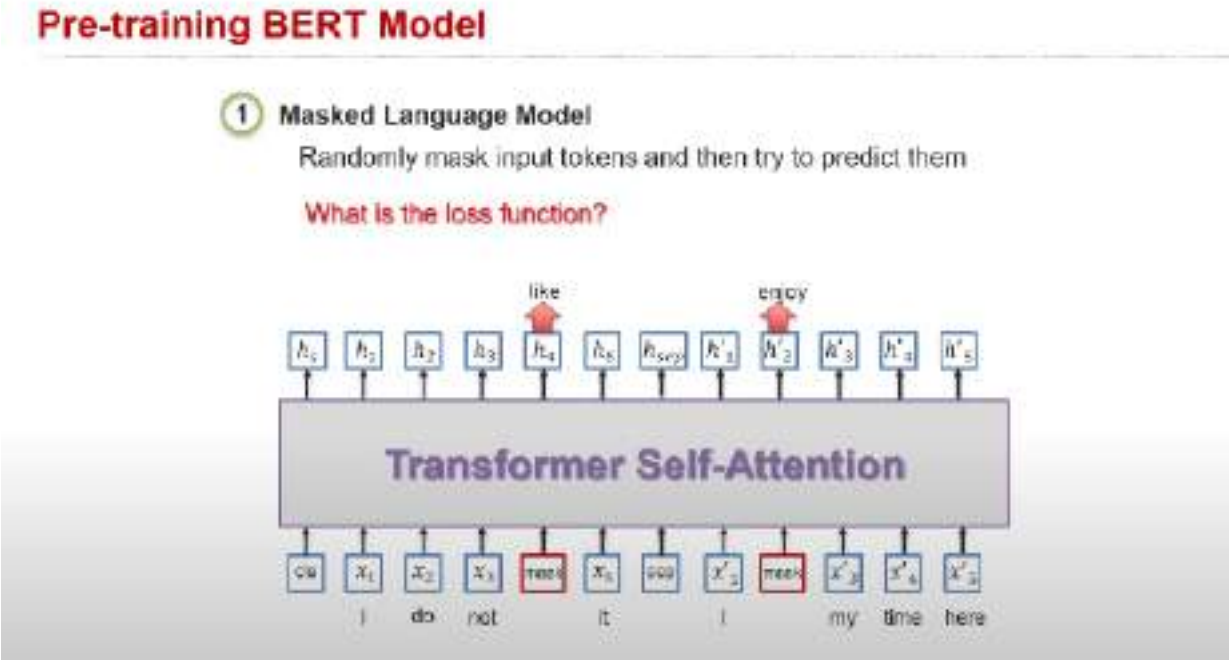
How does it help training deep neural networks?

For feedforward neural networks, training a deep network is usually very difficult, due to problems such as exploding gradients and vanishing gradients. On the other hand, the training process of a neural network with residual connections is empirically shown to converge much more easily, even if the network has several hundreds layers.





Bert bought masking along with a extra token for sentence ,
When we do masking and try to predict the next word , we do one hot encoding and the prediction is also one hot encoding but it can be a word embedding depending on the task .



To check if a sentence is next or not we have created a loss function called Isnext or notnext:

Pre-training BERT Model

② **Next Sentence Prediction**

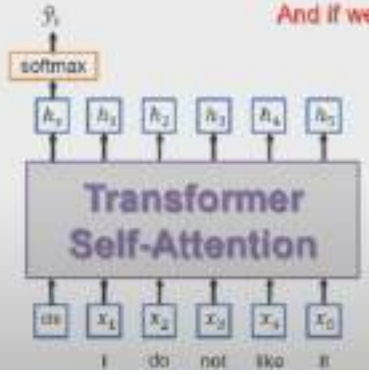
Given two sentences, predict if this is the next one or not

Fine-Tuning BERT

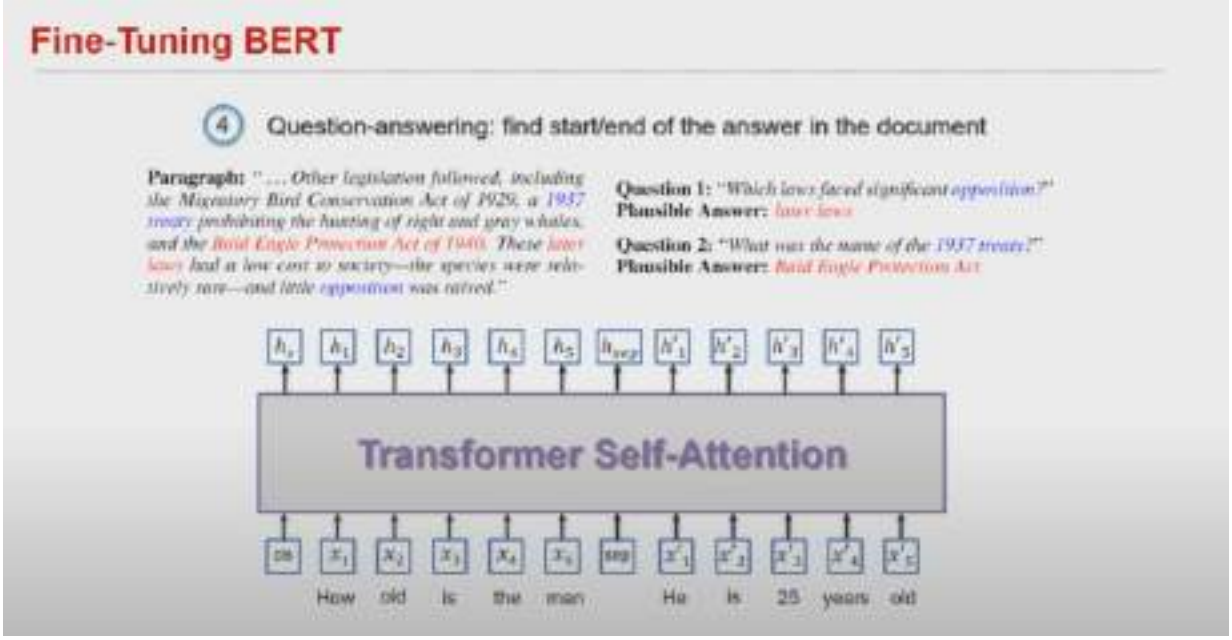
① **Sentence-level classification for only one sentence**

Examples: sentiment analysis, document classification

And if we have a label for each token?



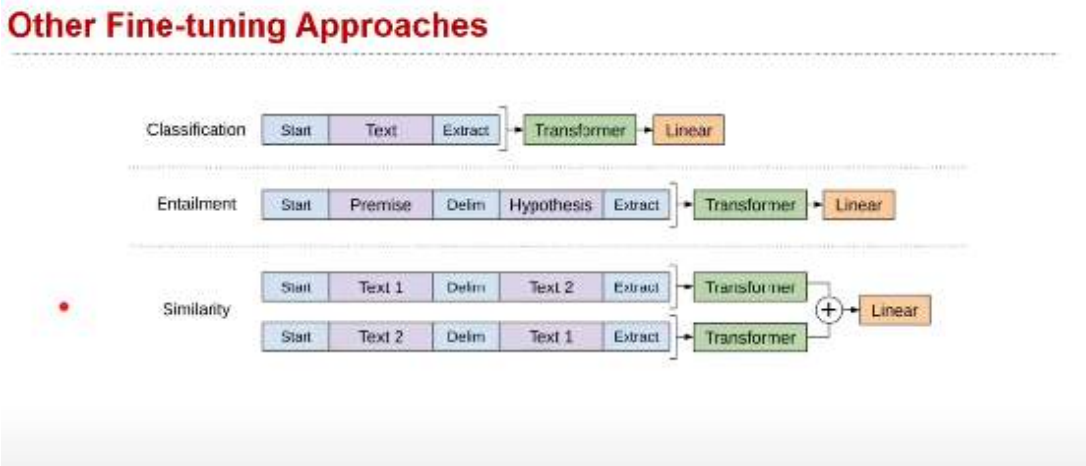
0

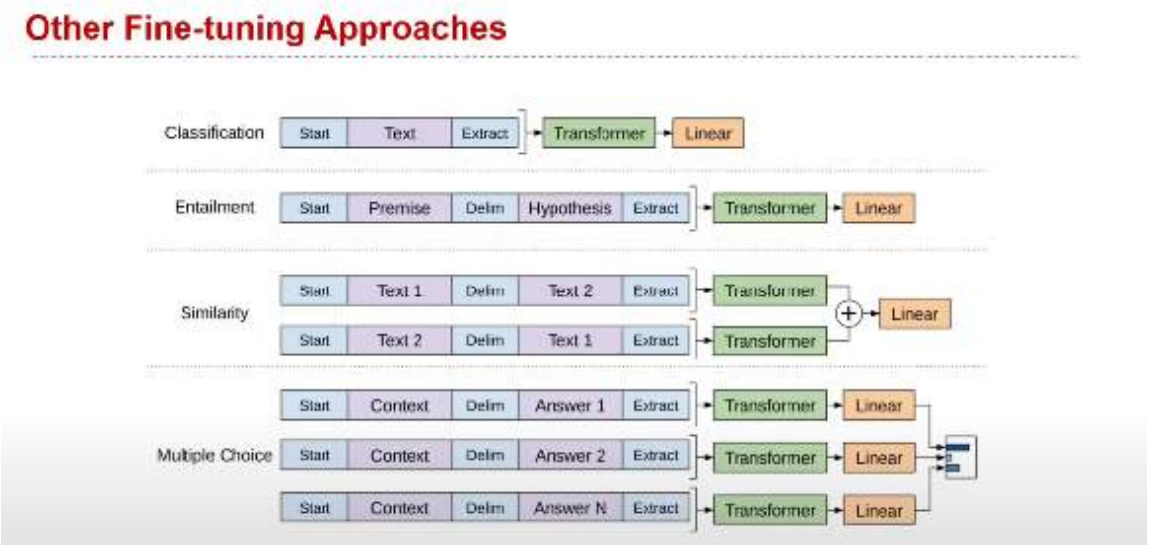


We need to find a good start to the answer to the question

Transformers is about contextualisation

We can do the following with Transformers :



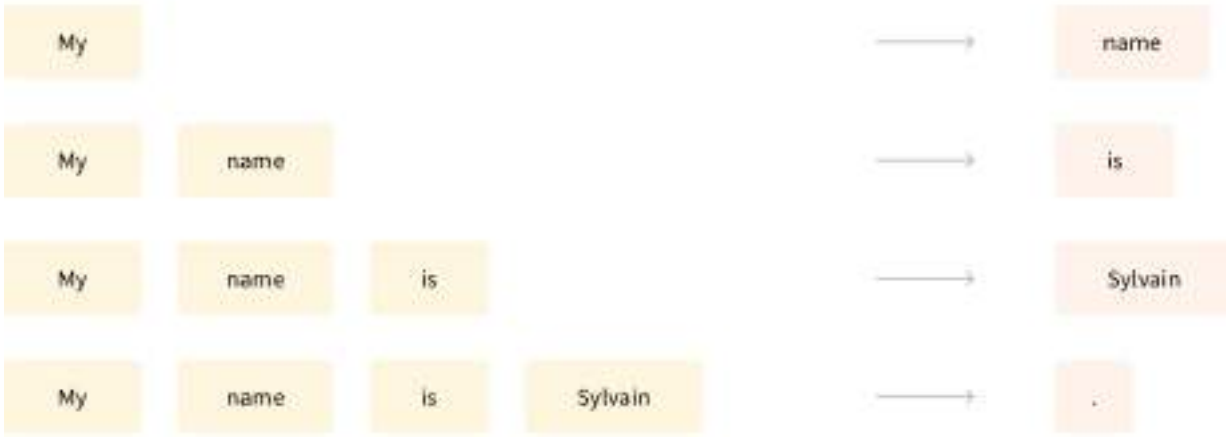


1.Text Classification Using Transformers:

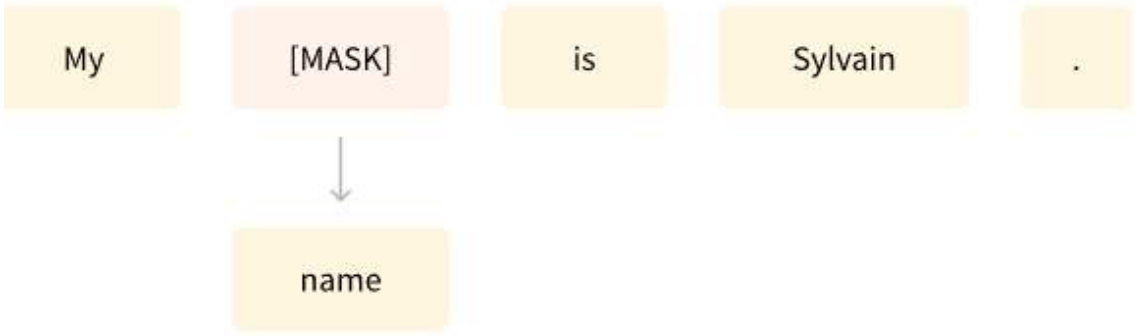
All the Transformer models mentioned above (GPT, BERT, BART, T5, etc.) have been trained as *language models*. This means they have been trained on large amounts of raw text in a self-supervised fashion. Self-supervised learning is a type of training in which the objective is automatically computed from the inputs of the model. That means that humans are not needed to label the data

This type of model develops a statistical understanding of the language it has been trained on, but it's not very useful for specific practical tasks. Because of this, the general pretrained model then goes through a process called *transfer learning*. During this process, the model is fine-tuned in a supervised way — that is, using human-annotated labels — on a given task

This type of model develops a statistical understanding of the language it has been trained on, but it's not very useful for specific practical tasks. Because of this, the general pretrained model then goes through a process called *transfer learning*. During this process, the model is fine-tuned in a supervised way — that is, using human-annotated labels — on a given task.
An example of a task is predicting the next word in a sentence having read the *n* previous words. This is called *causal language modeling* because the output depends on the past and present inputs, but not the future ones.

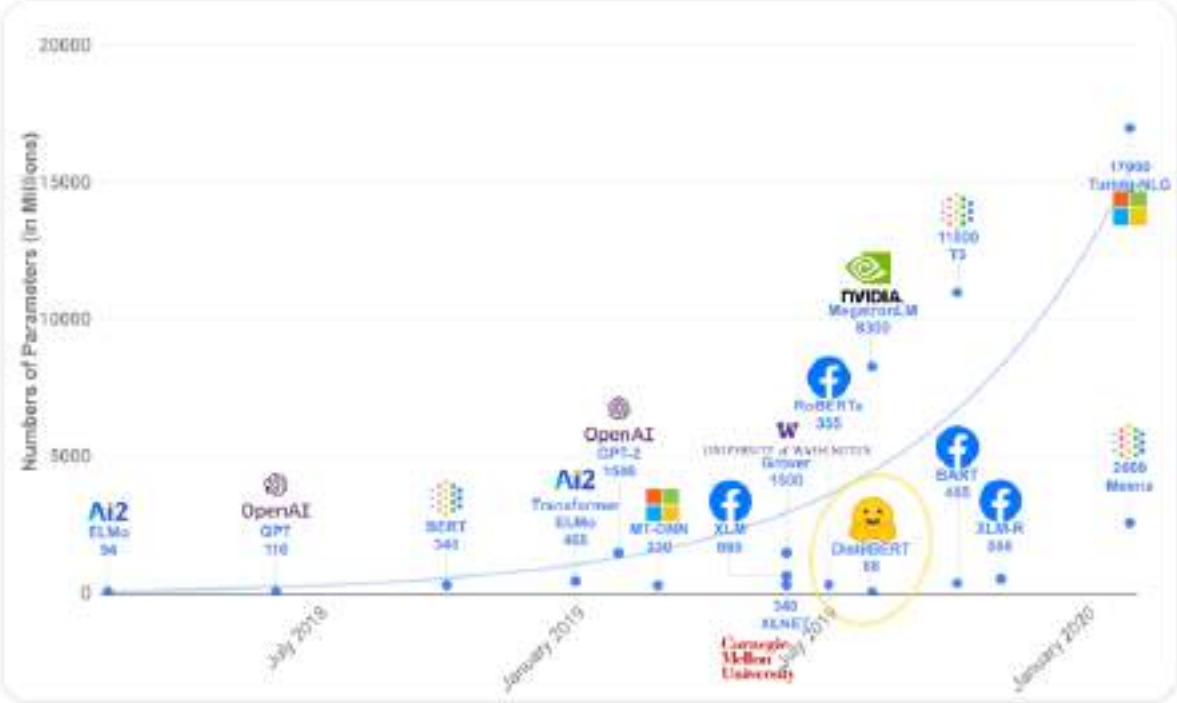


Another example is *masked language modeling*, in which the model predicts a masked word in the sentence



Transformers are big models:
Apart from a few outliers (like DistilBERT), the general strategy to achieve better performance is by increasing the models' sizes as well as the amount of data they are pretrained on.

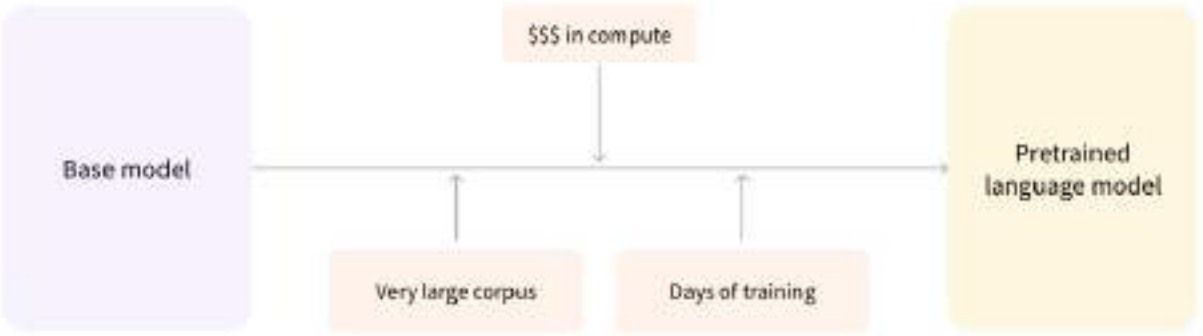
0



By the way, you can evaluate the carbon footprint of your models' training through several tools. For example ML CO2 Impact or Code Carbon which is integrated in 🤗 Transformers. To learn more about this, you can read this blog post which will show you how to generate an `emissions.csv` file with an estimate of the footprint of your training, as well as the documentation of 🤗 Transformers addressing this topic.

By the way, you can evaluate the carbon footprint of your models' training through several tools. For example ML CO2 Impact or Code Carbon which is integrated in 🤗 Transformers. To learn more about this, you can read this blog post which will show you how to generate an `emissions.csv` file with an estimate of the footprint of your training, as well as the documentation of 🤗 Transformers addressing this topic.

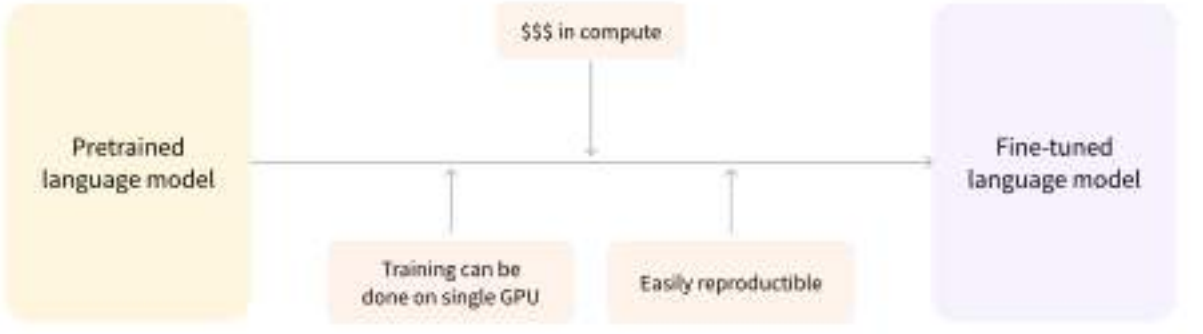
Pretraining is the act of training a model from scratch: the weights are randomly initialized, and the training starts without any prior knowledge.



Fine-tuning, on the other hand, is the training done after a model has been pretrained. To perform fine-tuning, you first acquire a pretrained language model, then perform additional training with a dataset specific to your task.
why not simply train directly for the final task:

> The pretrained model was already trained on a dataset that has some similarities with the fine-tuning dataset. The fine-tuning process is thus able to take advantage of knowledge acquired by the initial model during pretraining (for instance, with NLP problems, the pretrained model will have some kind of statistical understanding of the language you are using for your task).
>
> Since the pretrained model was already trained on lots of data, the fine-tuning requires way less data to get decent results.
>
> For the same reason, the amount of time and resources needed to get good results are much lower.

For example, one could leverage a pretrained model trained on the English language and then fine-tune it on an arXiv corpus, resulting in a science/research-based model. The fine-tuning will only require a limited amount of data: the knowledge the pretrained model has acquired is "transferred," hence the term *transfer learning*.

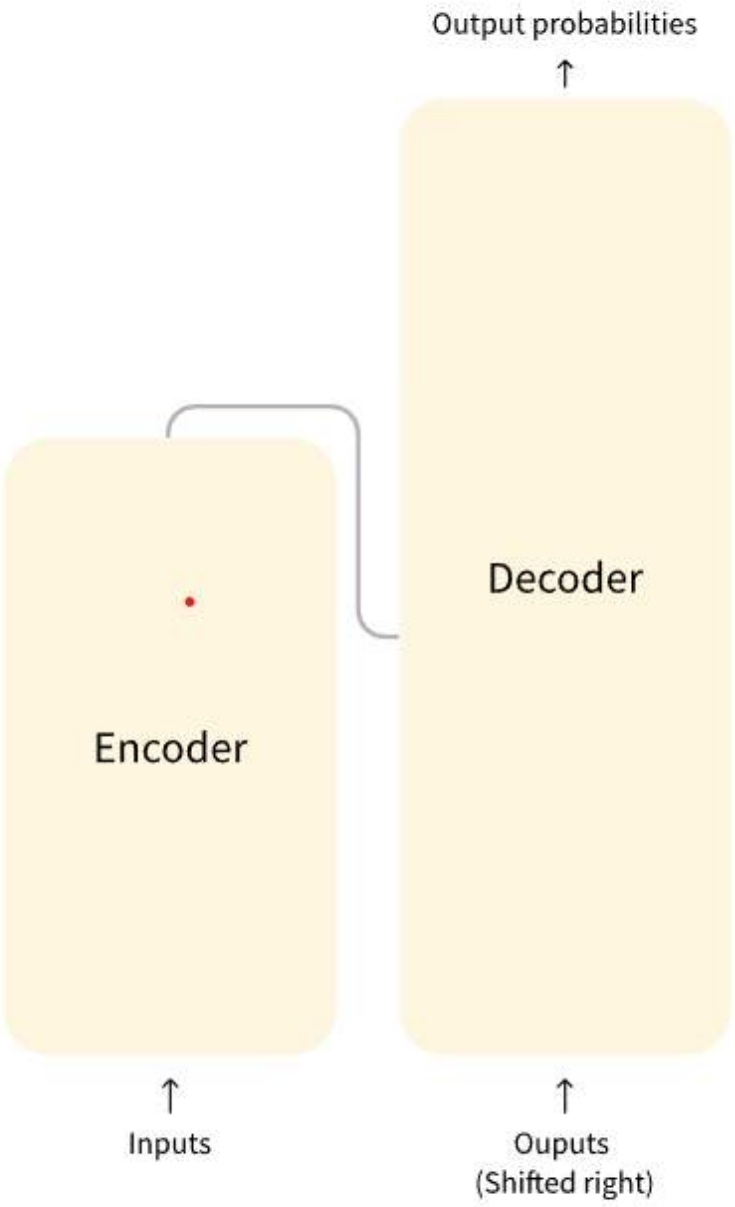


Attention Layers:

A key feature of Transformer models is that they are built with special layers called *attention layers,*this layer will tell the model to pay specific attention to certain words in the sentence you passed it (and more or less ignore the others) when dealing with the representation of each word.

The model is primarily composed of two blocks:

- Encoder (left): 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.
- Decoder (right): 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.



Each of these parts can be used independently, depending on the task:

- **Encoder-only models**: Good for tasks that require understanding of the input, such as sentence classification and named entity recognition.
- **Decoder-only models**: Good for generative tasks such as text generation.
- **Encoder-decoder models** or **sequence-to-sequence models**: Good for generative tasks that require an input, such as translation or summarization.

To put this into context, consider the task of translating text from English to French. Given the input "You like this course", a translation model will need to also attend to the adjacent word "You" to get the proper translation for the word "like", because in French the verb "like" is conjugated differently depending on the subject. The rest of the sentence, however, is not useful for the translation of that word. In the same vein, when translating "this" the model will also need to pay attention to the word "course", because "this" translates differently depending on whether the associated noun is masculine or feminine. Again, the other words in the sentence will not matter for the translation of "this". With more complex sentences (and more complex grammar rules), the model would need to pay special attention to words that might appear farther away in the sentence to properly translate each word.

The same concept applies to any task associated with natural language: a word by itself has a meaning, but that meaning is deeply affected by the context, which can be any other word (or words) before or after the word being studied.

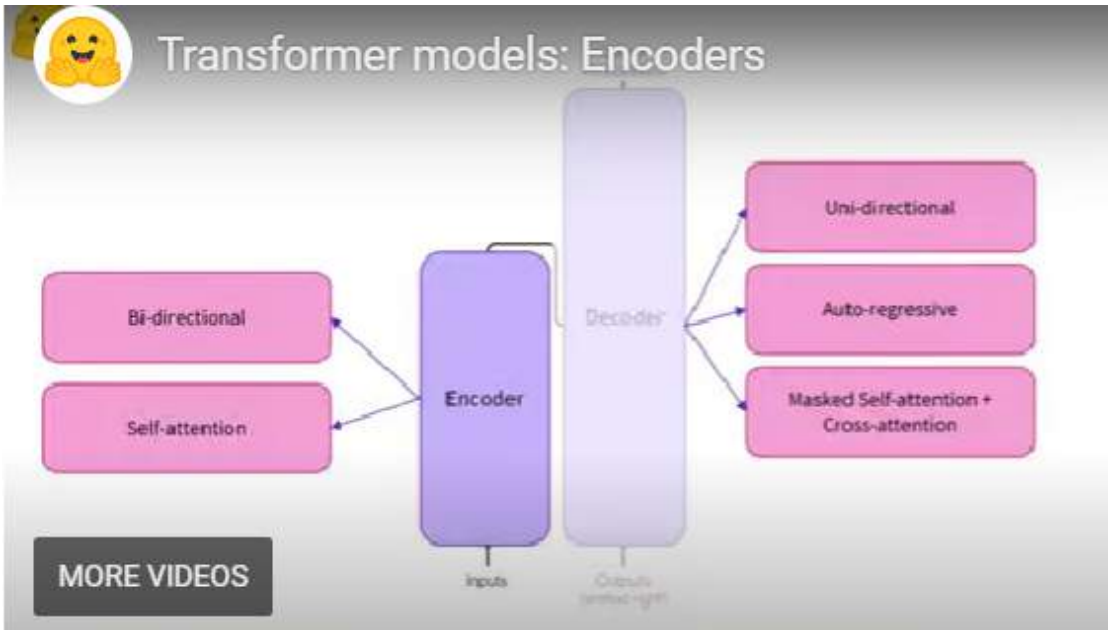
The original Architecture :

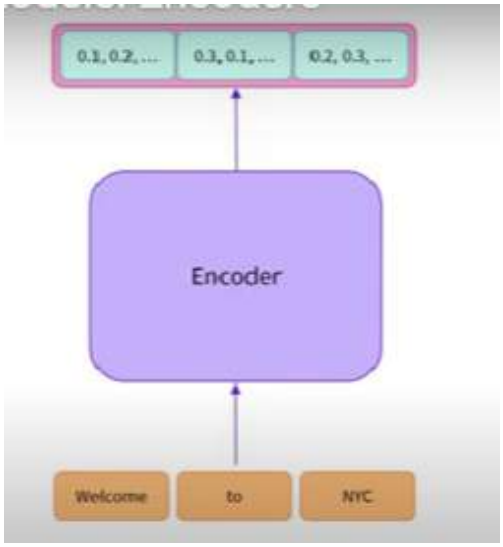
The Transformer architecture was originally designed for translation. During training, the encoder receives inputs (sentences) in a certain language, while the decoder receives the same sentences in the desired target language

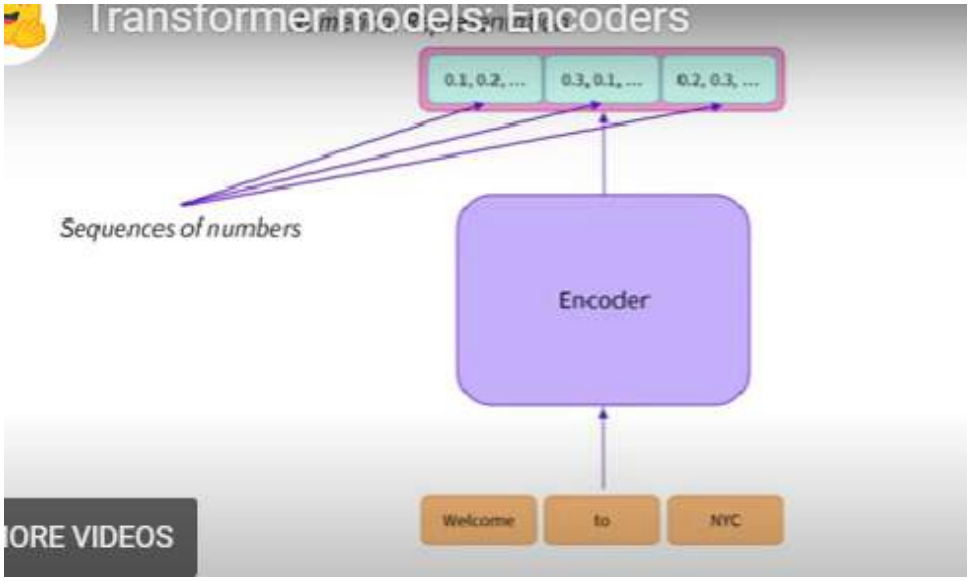
The original architecture

The Transformer architecture was originally designed for translation. During training, the encoder receives inputs (sentences) in a certain language, while the decoder receives the same sentences in the desired target language. In the encoder, the attention layers can use all the words in a sentence (since, as we just saw, the translation of a given word can be dependent on what is after as well as before it in the sentence). The decoder, however, works sequentially and can only pay attention to the words in the sentence that it has already translated (so, only the words before the word currently being generated). For example, when we have predicted the first three words of the translated target, we give them to the decoder which then uses all the inputs of the encoder to try to predict the fourth word.

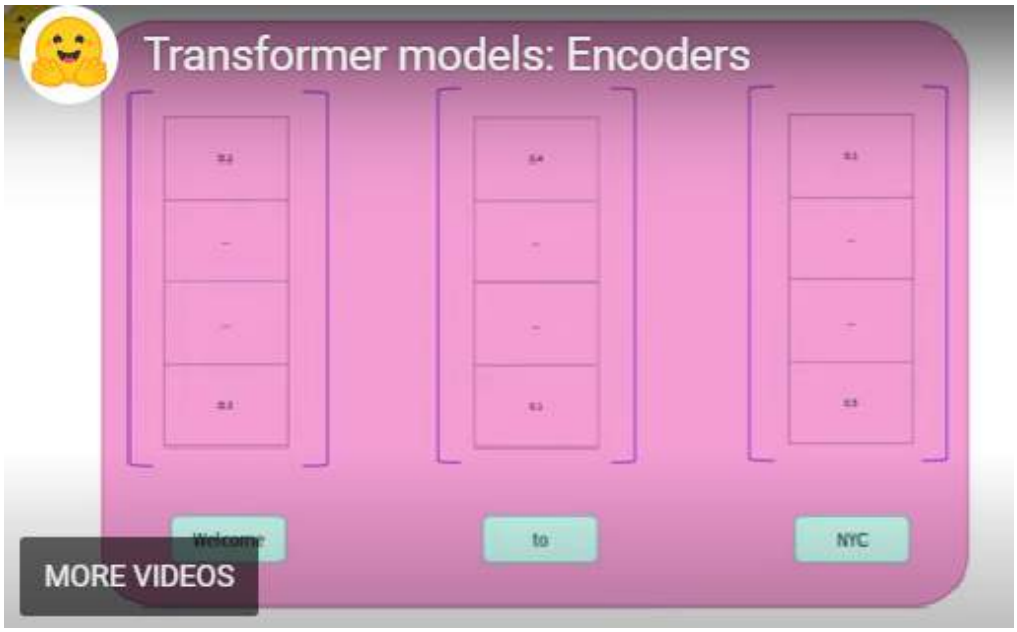
To speed things up during training (when the model has access to target sentences), the decoder is fed the whole target, but it is not allowed to use future words (if it had access to the word at position 2 when trying to predict the word at position 2, the problem would not be very hard!). For instance, when trying to predict the fourth word, the attention layer will only have access to the words in positions 1 to 3.



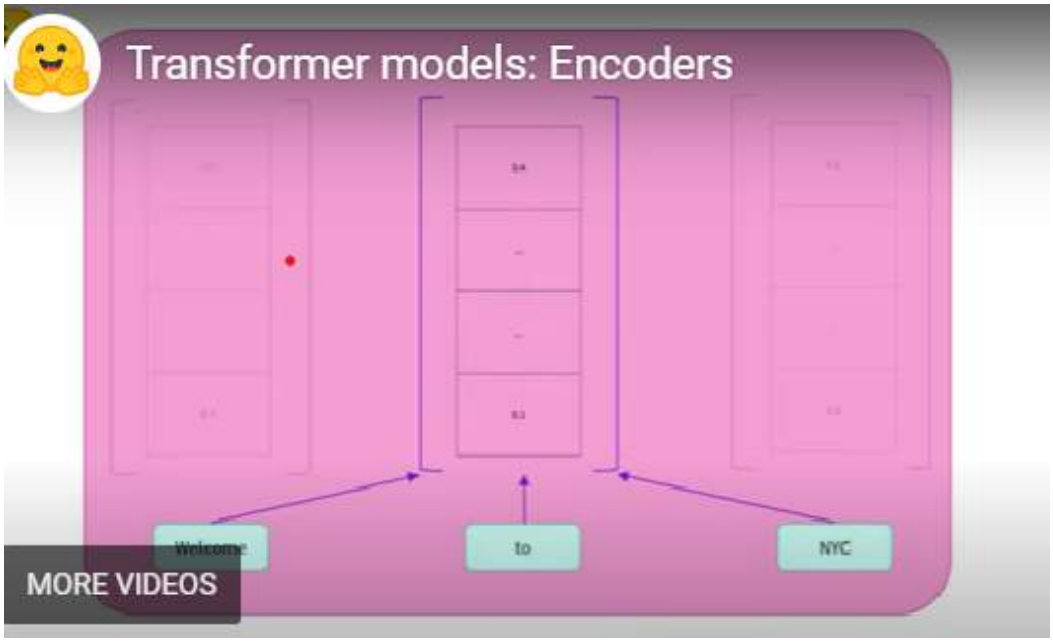




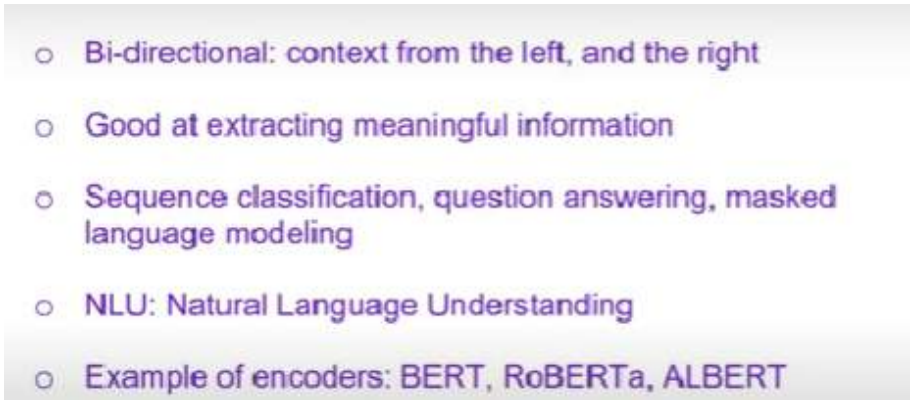
The bert can 768 word or token ans the length will be as shown below :



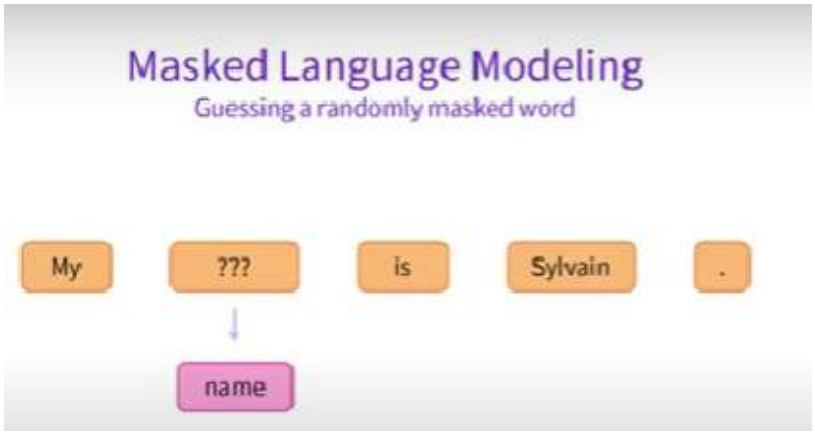
Each word also takes into account the words around it :



This happens because of the self attention mechanism,Bert is stand alone encoder model, it can be used for question answering



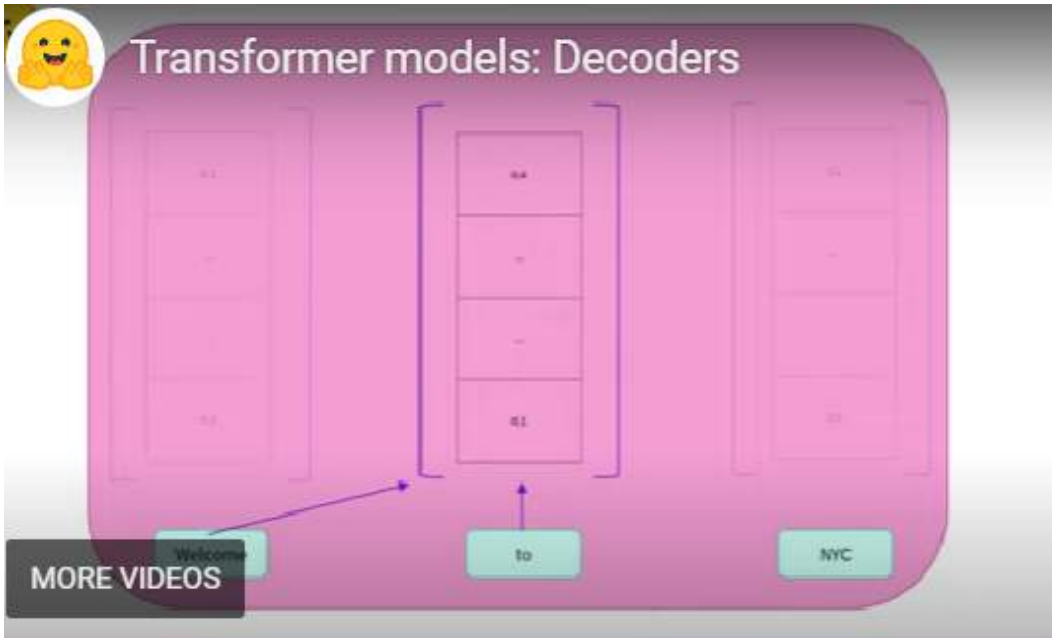
Following are Encoder only models:



Encoder models are used for sentiment analysis as well .

Decoder Architecture:

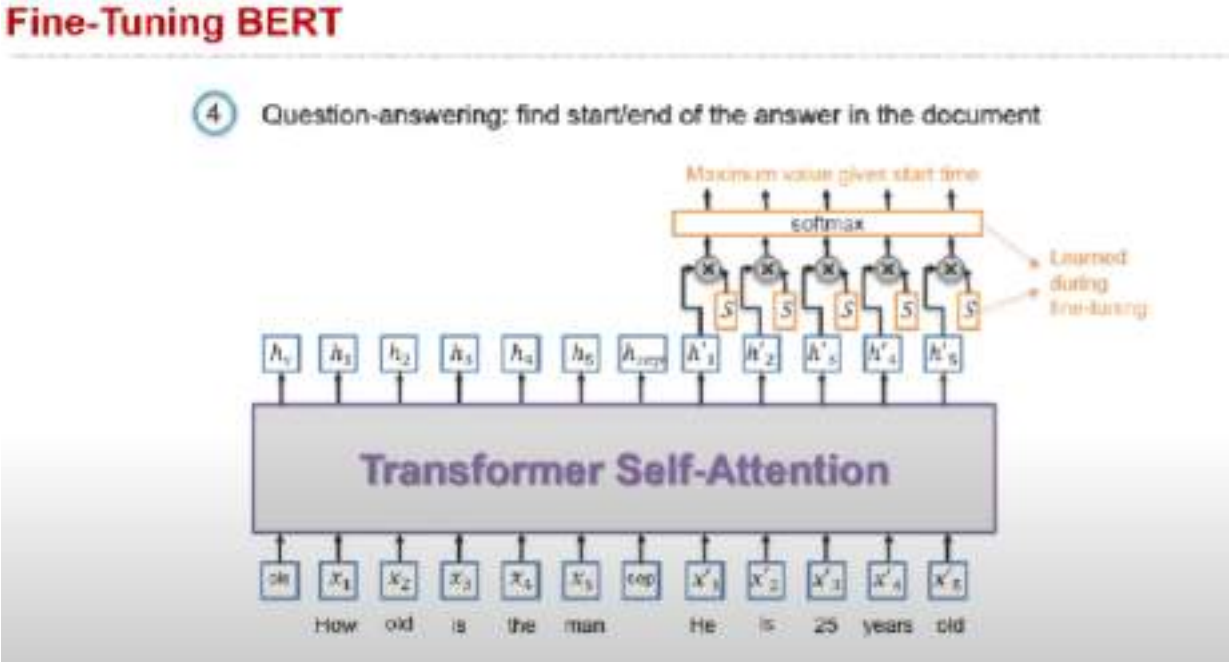
The decoder is uni directional as shown below :



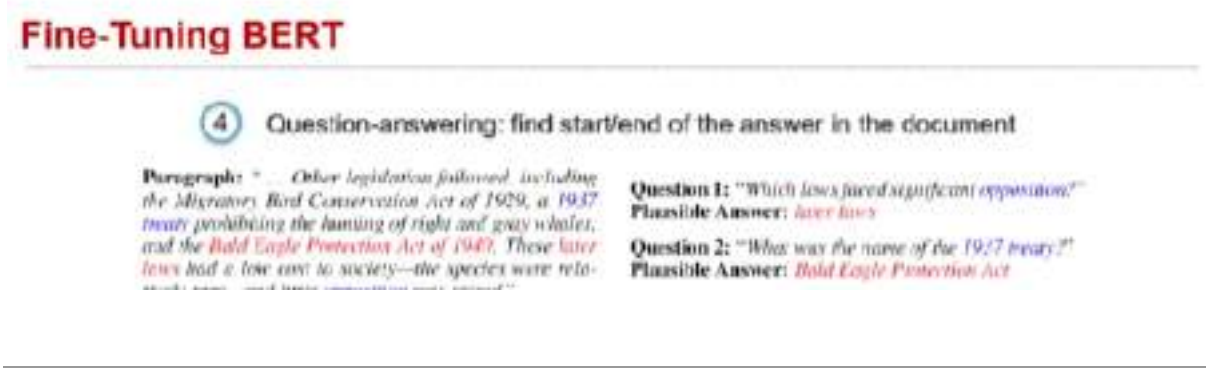
The words on the right are masked the decoder model has masked self attention mechanism

- Unidirectional: access to their left (or right!) context
- Great at causal tasks; generating sequences
- NLG: Natural Language generation
- Example of decoders: GPT-2, GPT Neo

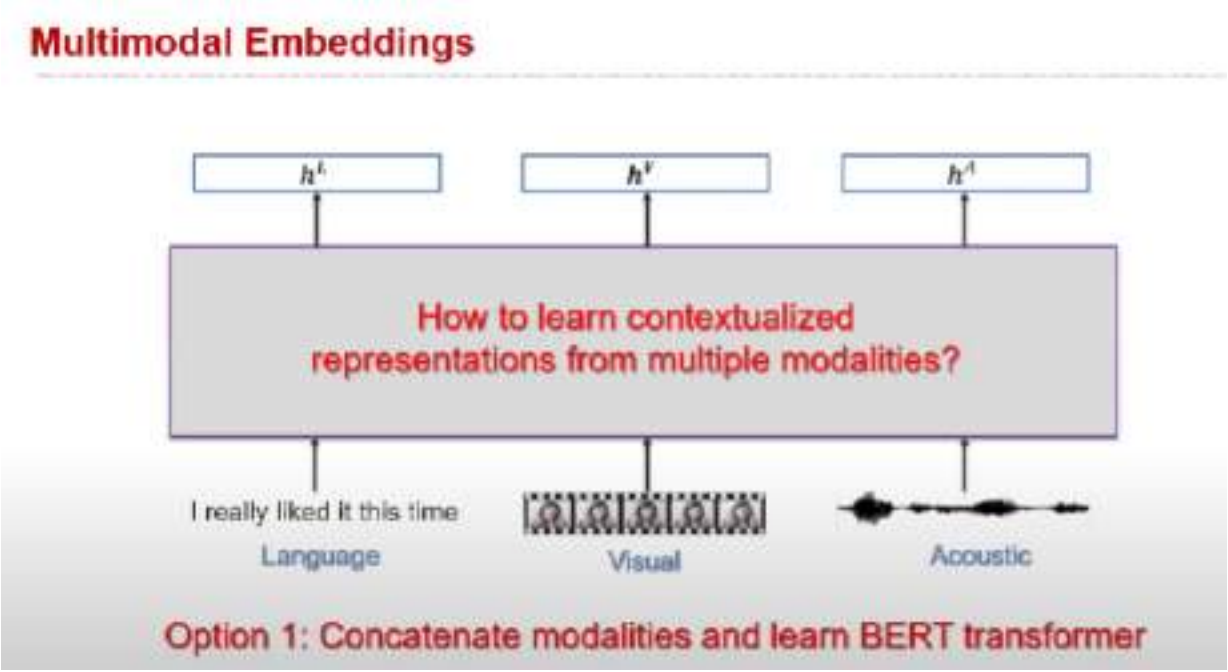
Back to Multi Modal Question Answering:
The bert is fine tuned in the following manner :

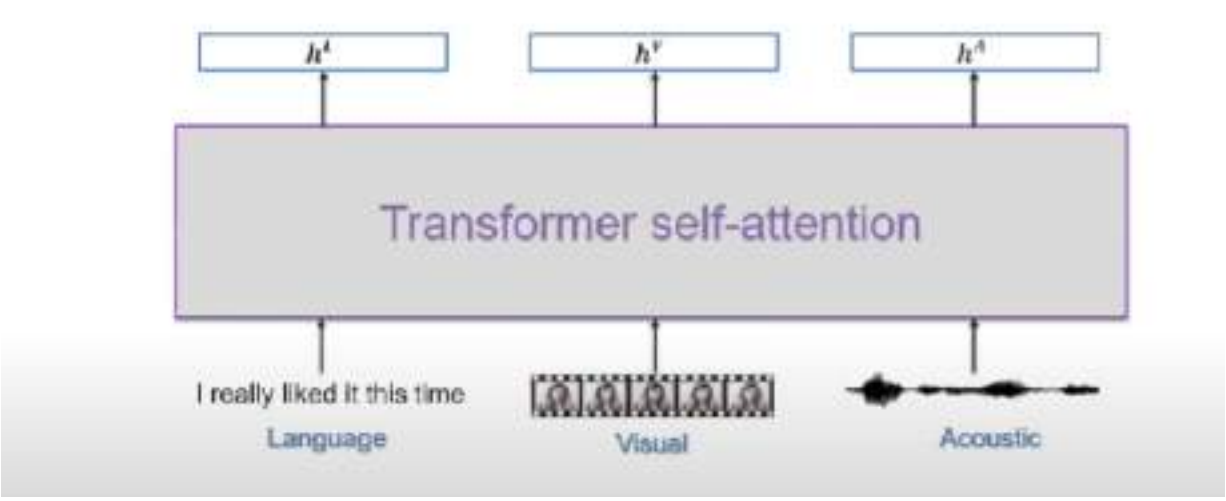


In this we are seeing which word would be a good start to a question we have asked as shown below:



Multi Modal Bert

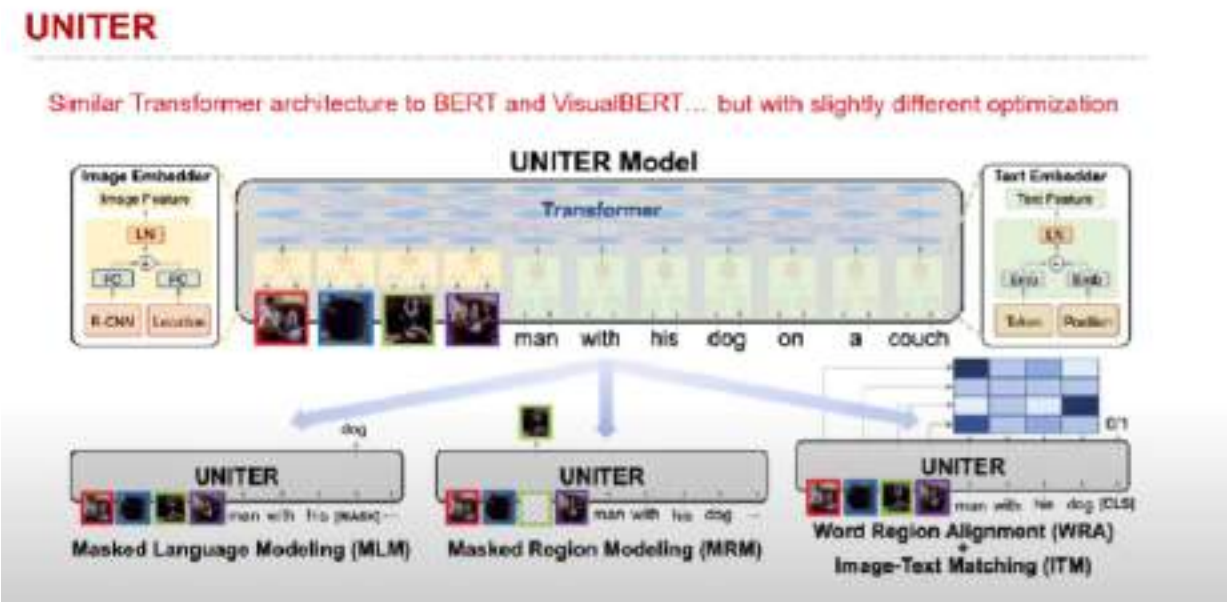




The visual bert has extra positional embedding which basically says that if the given tensor is a image token or word token:

In visual bert the words are sent into the embedding space but even small parts of the images are fed into the transformer as embeddings

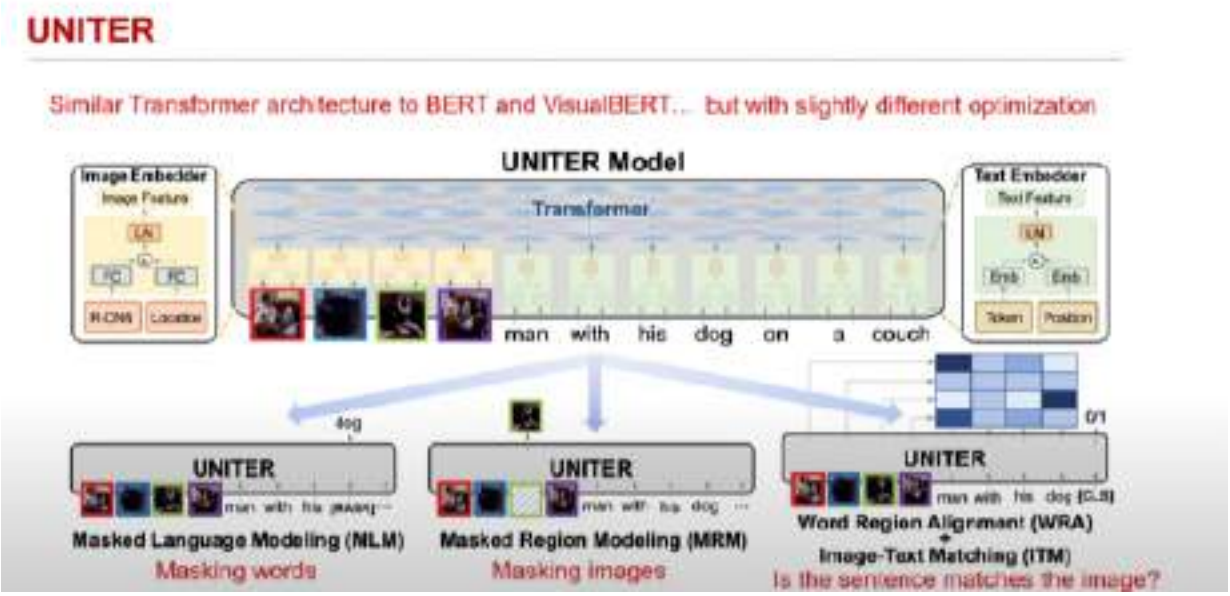
A better version than the bert and visual bert are the limiter model which has the three losses as shown below :



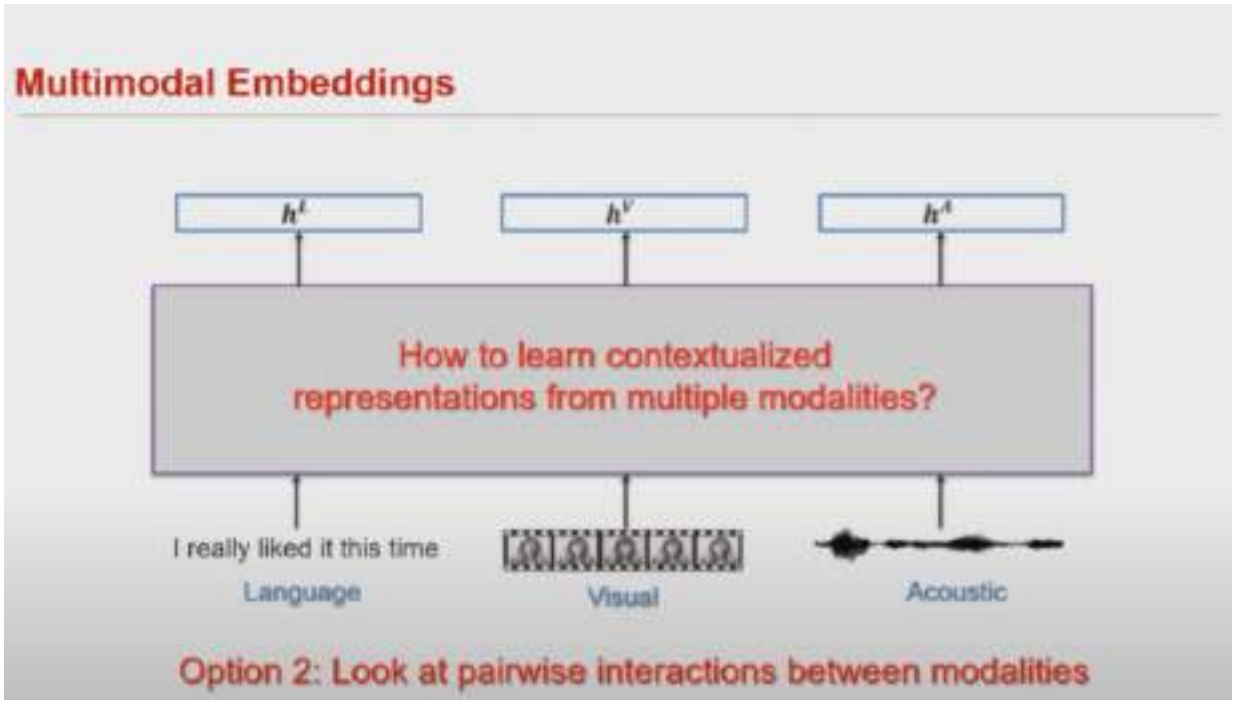
L1 :Masked Language Modelling : This basically masks a word in the input and see how it performs

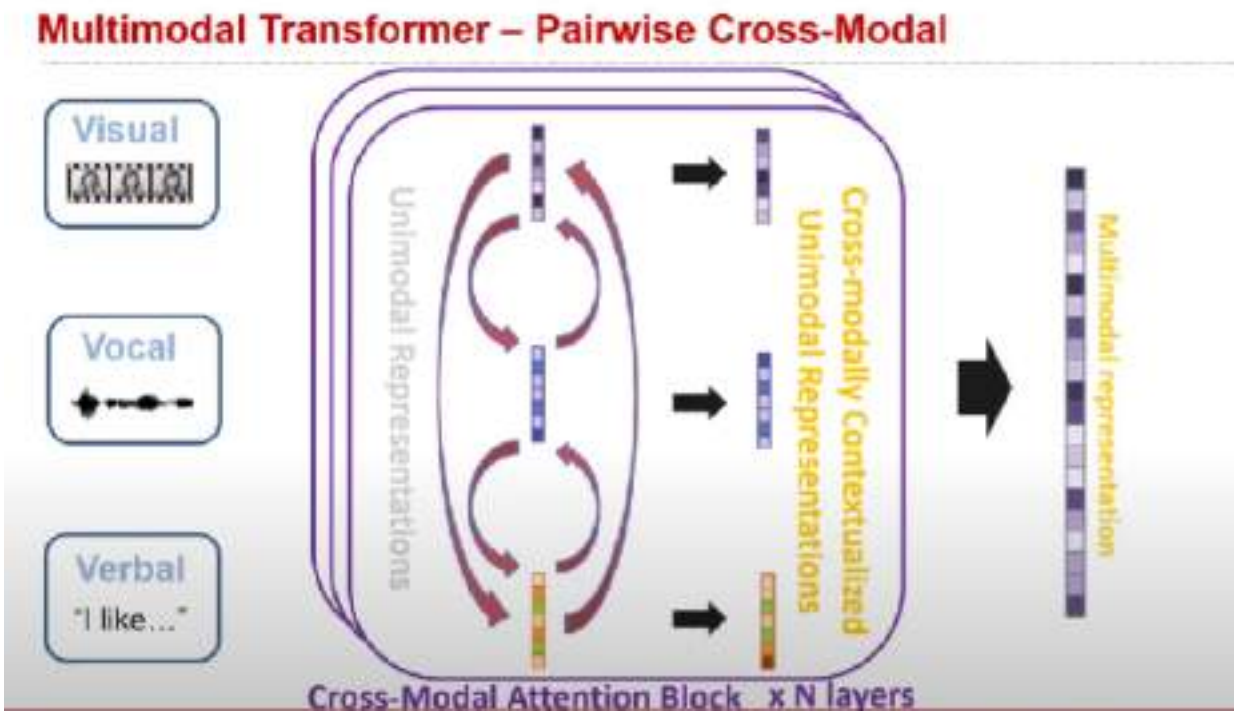
L2: Masked Region Modelling: This masks a region and does the same

The above is Early fusion,



We also have the pair wise way , which means we have 6 transformers doing 2 at a time :

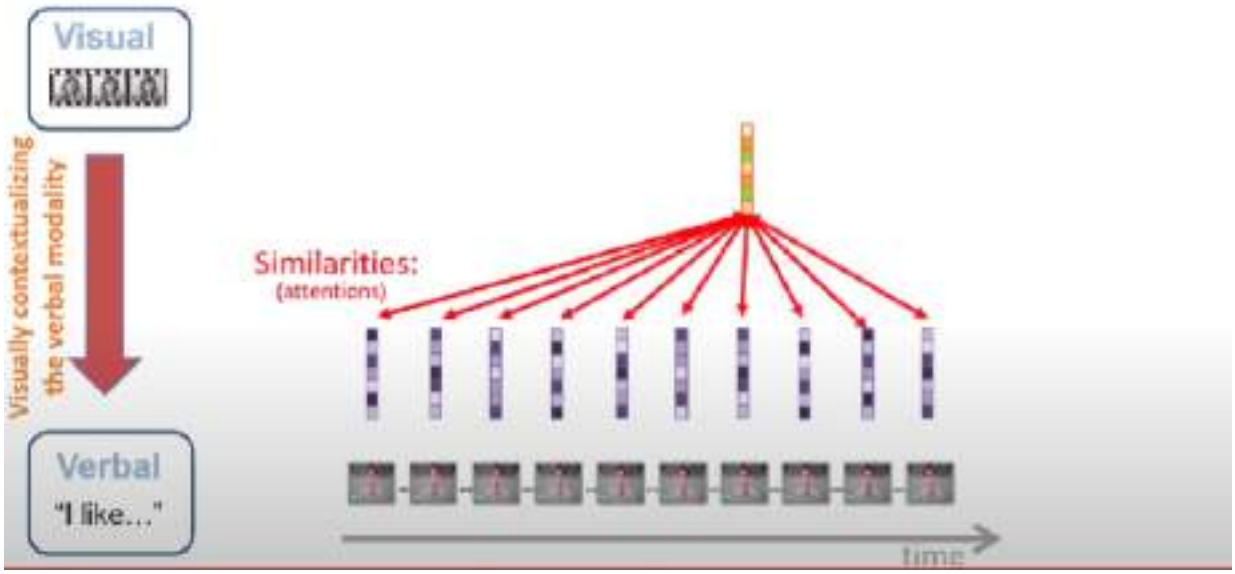




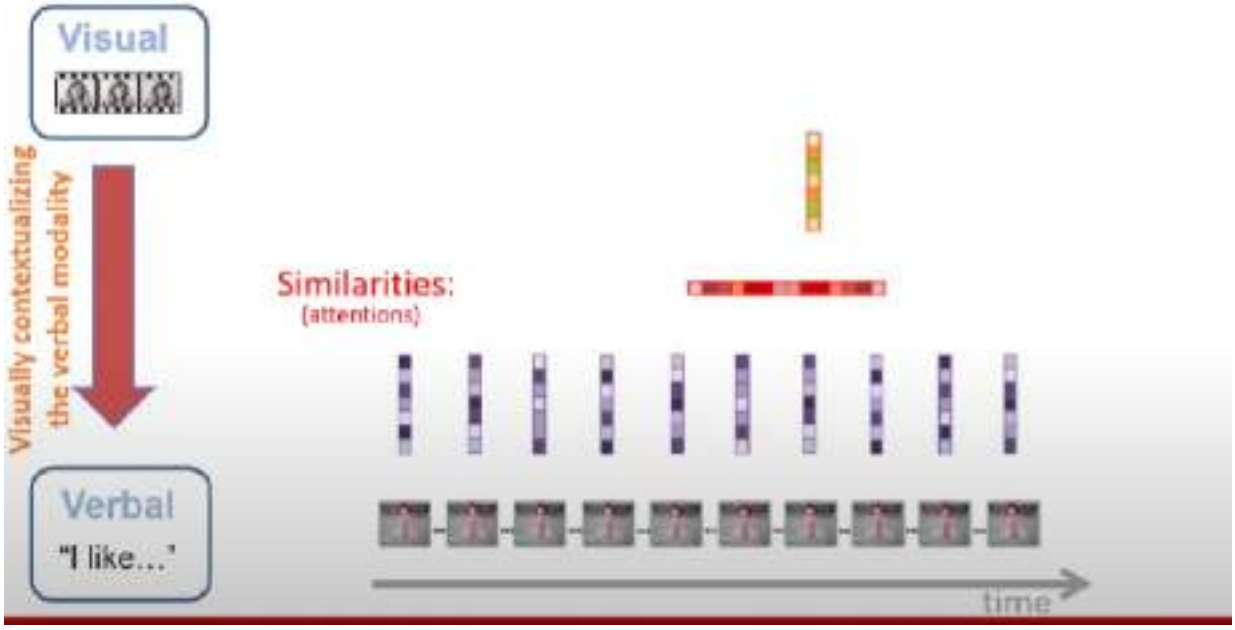
In pairwise what happens is that we dont have one transformer but we have 6 transformers:
one takes language and image and so on
Attention is used in the same way in this case between the different modalities
One modalitiy will have different way of being contextualised with other modality

for example language will have method A with image while it will have method B with audio
This is how it works :
Attention is done between the two modalities as shown below
we have a word and we see which part of the image is related to it by calculating the attention weight which is then multiplied by the visual embedding as shown below:

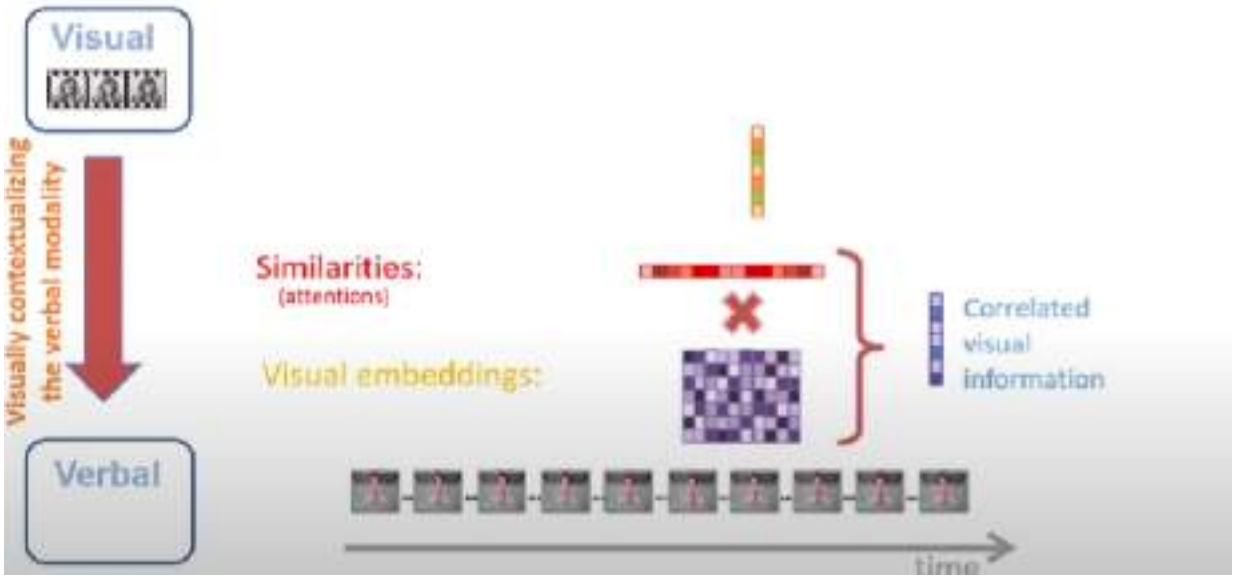
Cross-Modal Transformer Module ($V \rightarrow L$)



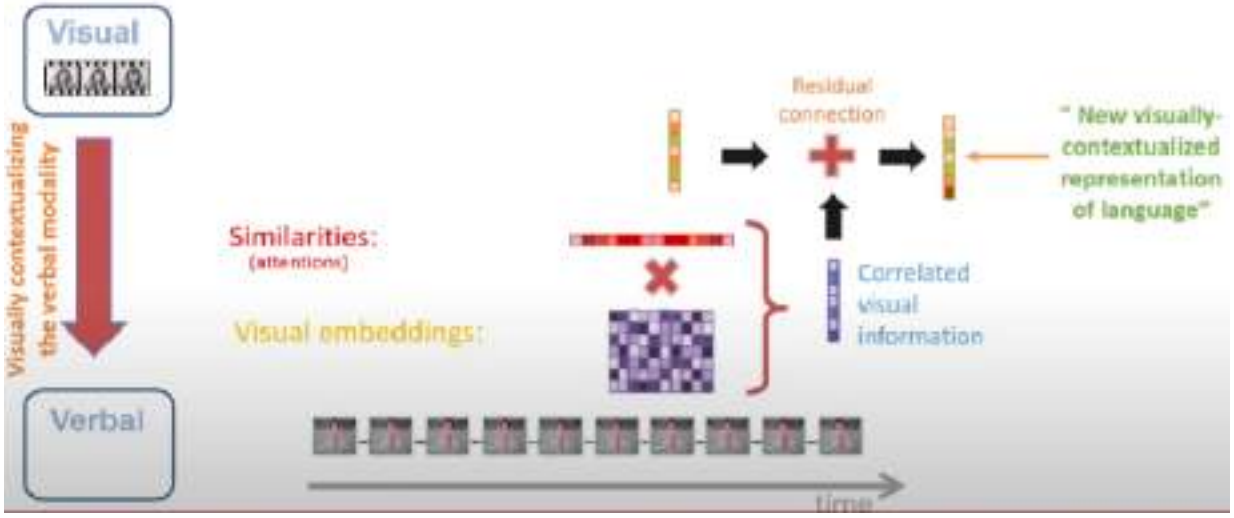
Cross-Modal Transformer Module ($V \rightarrow L$)

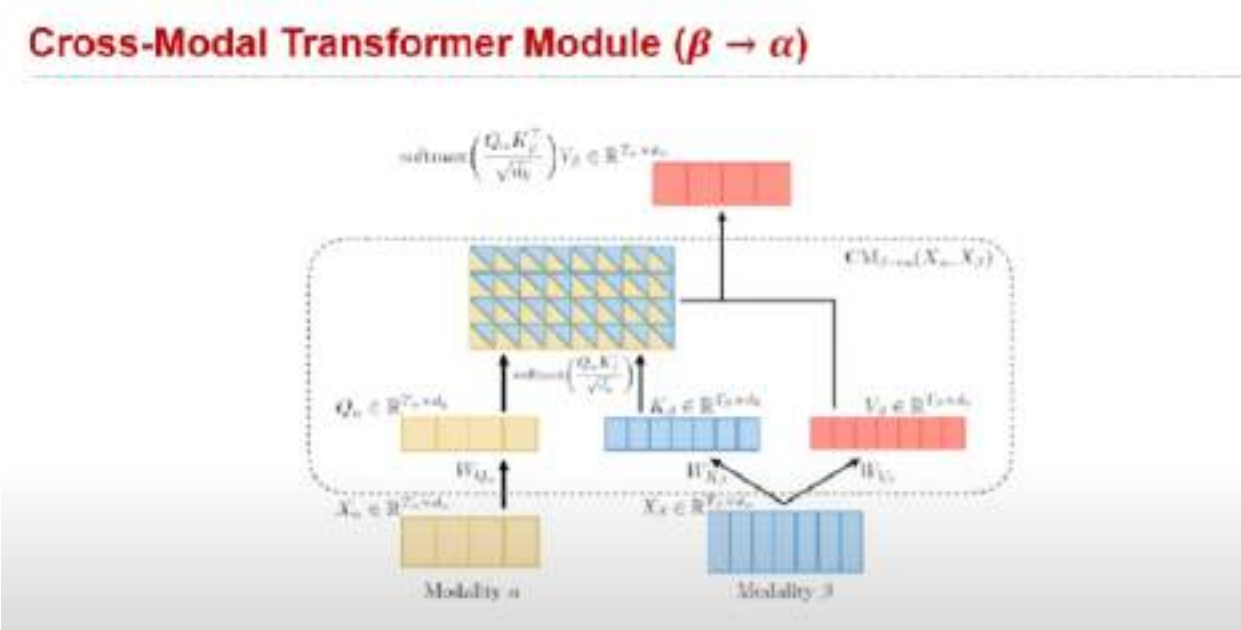


Cross-Modal Transformer Module ($V \rightarrow L$)



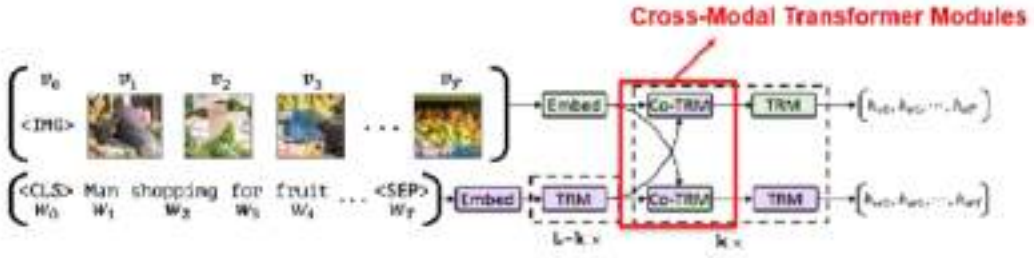
Cross-Modal Transformer Module ($V \rightarrow L$)



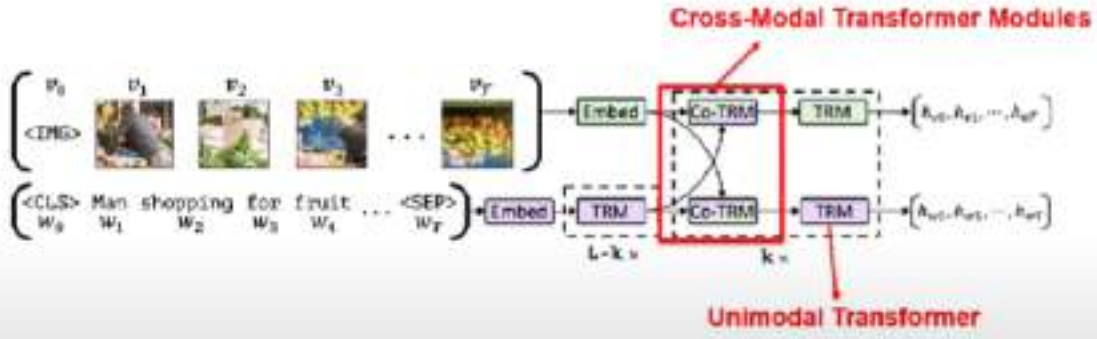


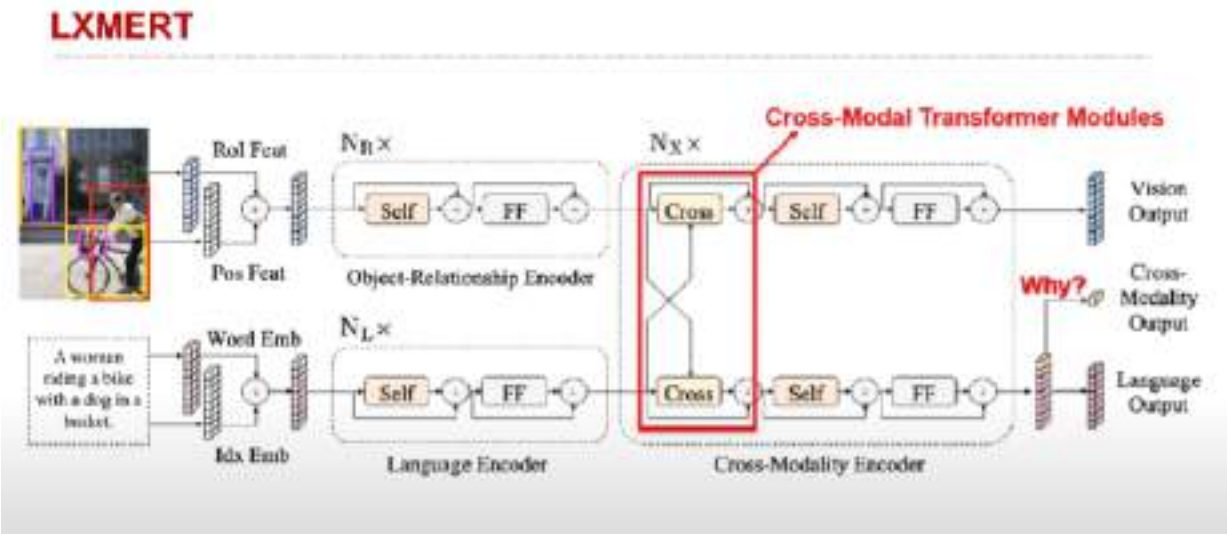
The above can be used to make vilbert (visual + language):
Vilbert is cross modal transformer

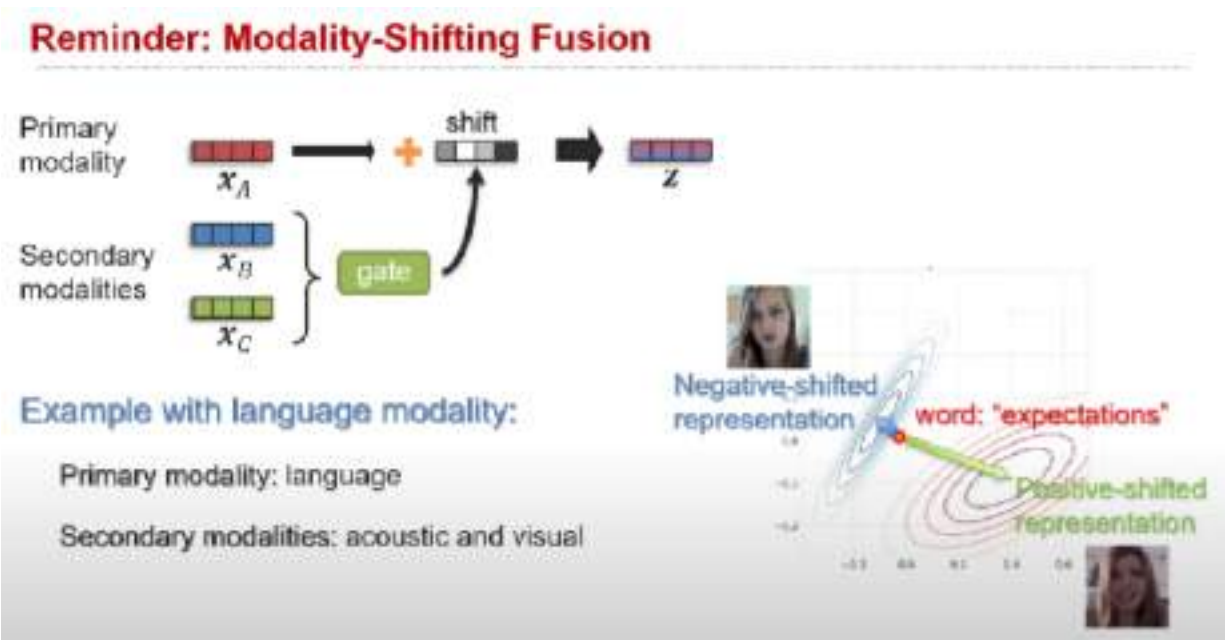
ViLBERT



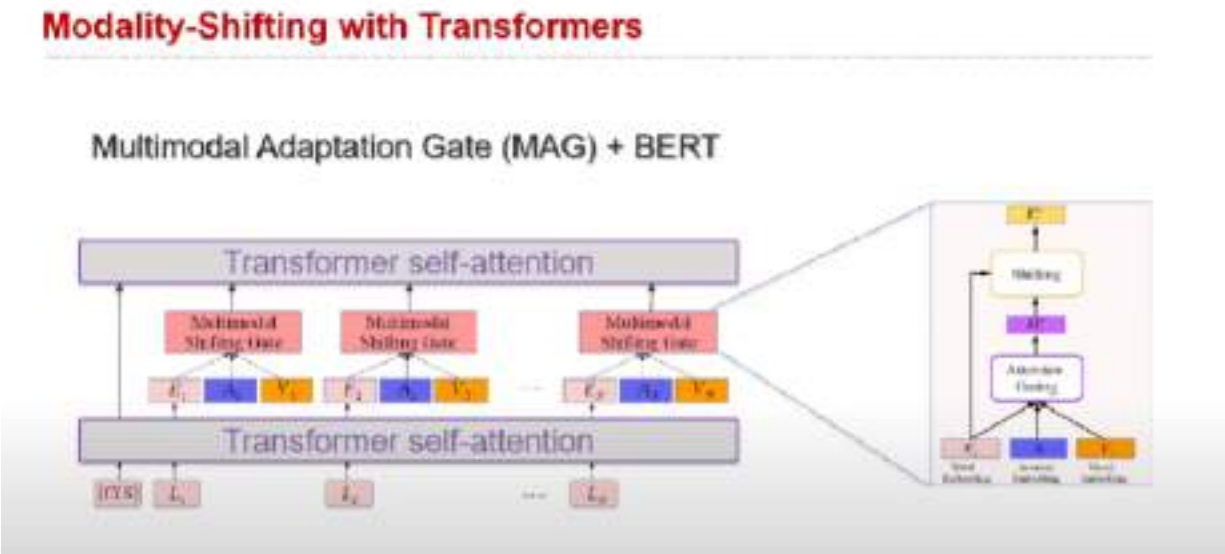
ViLBERT







Shift means that the certain percentage of secondary modality



Vision Transformers:

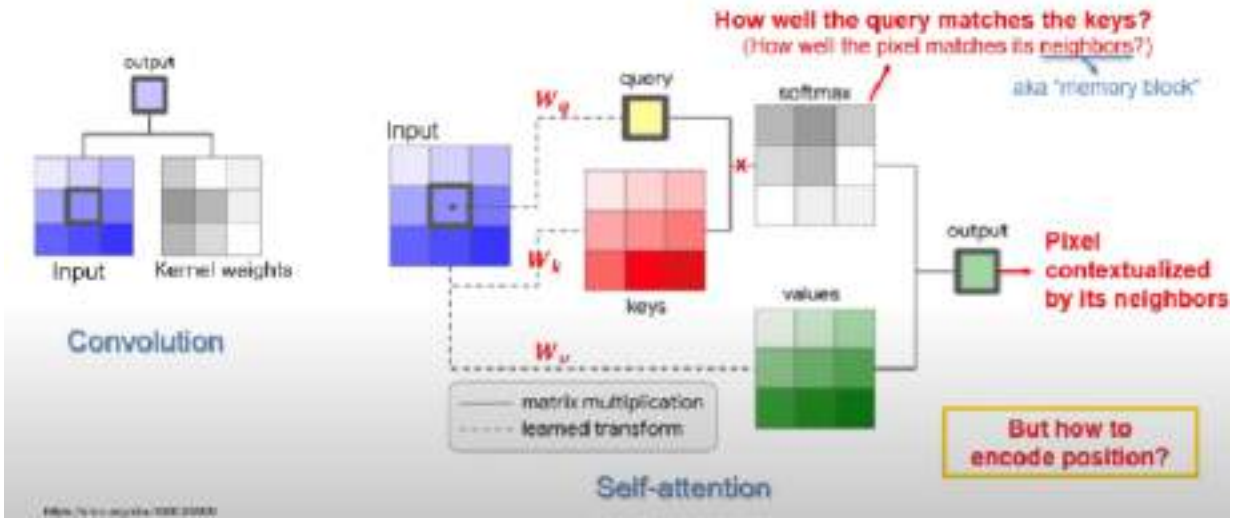
No more CNN in it :

Transformers are good at contextualising and we can apply the same fundamentals by contextualising one pixel with its neighbour and so on.

This is done with the help of self attention

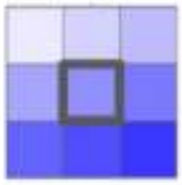
To care of the position we make use of the fact of relative embedding

Replacing a CNN w/ Self-Attention



Replacing a CNN w/ Self-Attention

Image patch



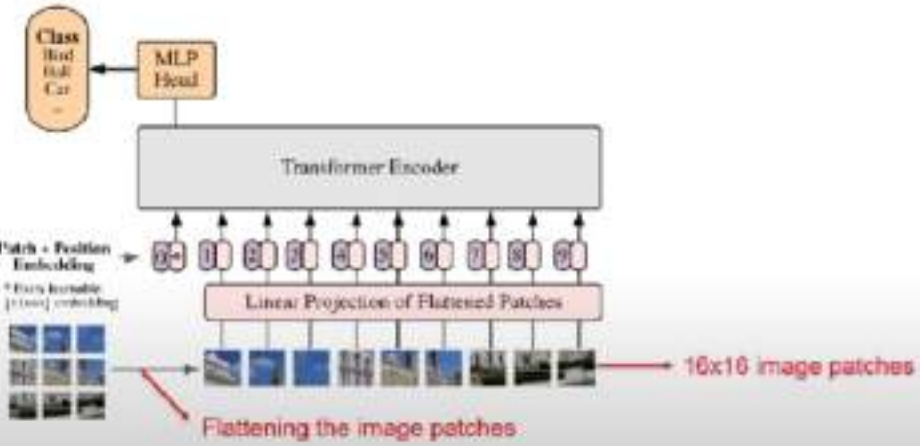
2D relative position embedding

-1, -1	-1, 0	-1, 1	-1, 2
0, -1	0, 0	0, 1	0, 2
1, -1	1, 0	1, 1	1, 2
2, -1	2, 0	2, 1	2, 2

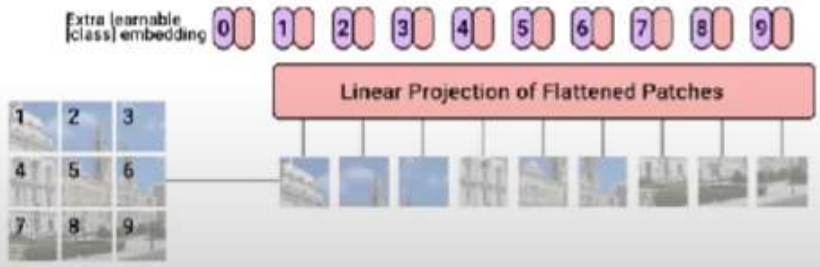
Position embedding is added to the key:

$$y_{ij} = \sum_{a,b \in \mathcal{N}_k(i,j)} \mathtt{softmax}_{ab} \left(q_{ij}^\top k_{ab} + q_{ij}^\top r_{a-i,b-j} \right) v_{ab}$$

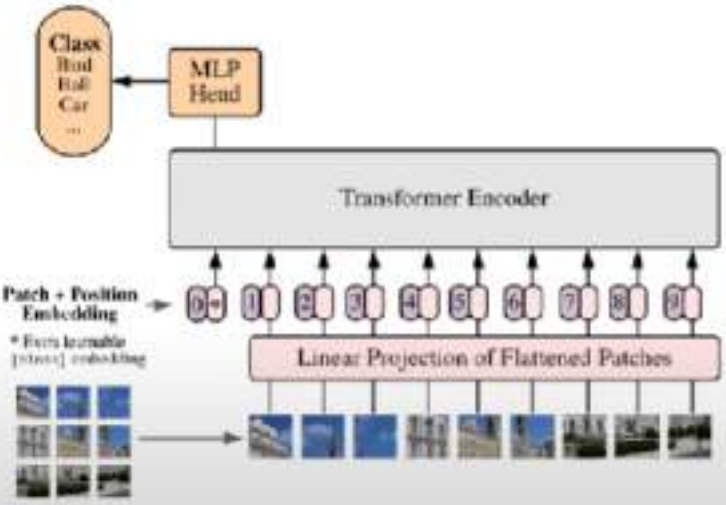
Vision Transformer (ViT)



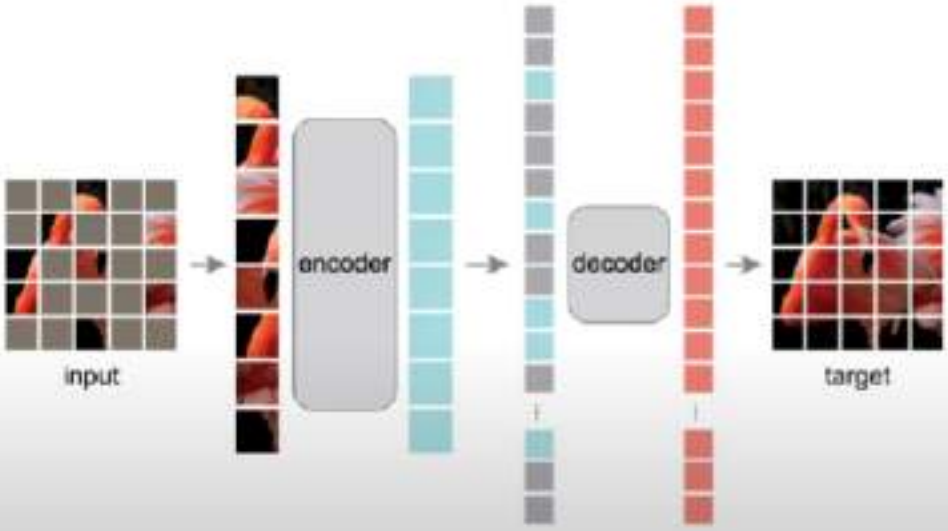
Vision Transformer (ViT)



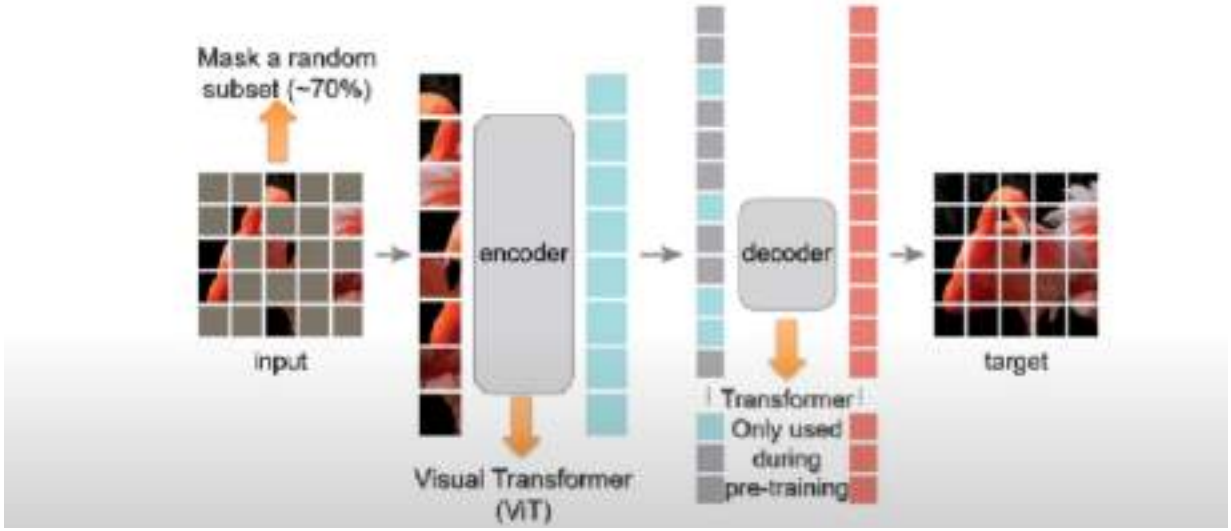
Vision Transformer (ViT)



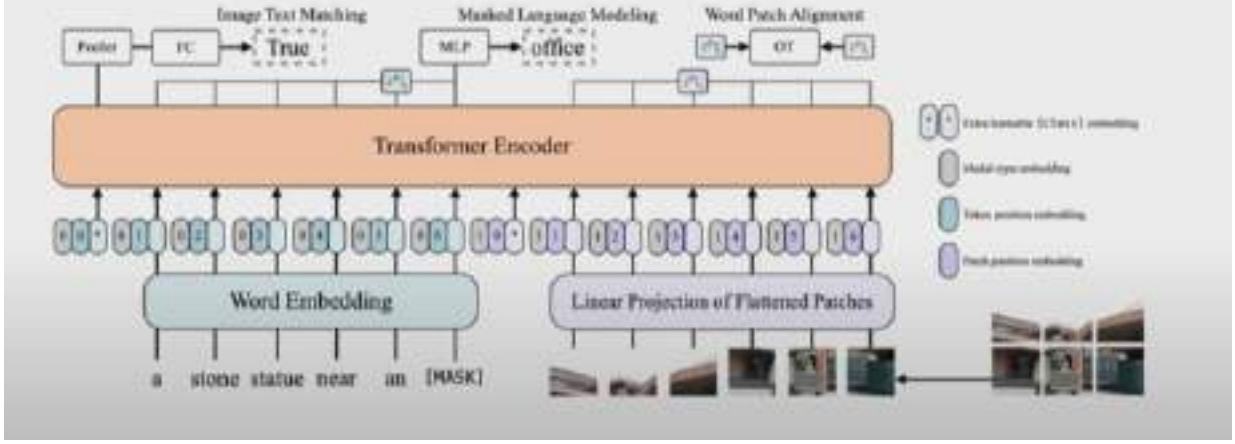
Masked Auto-Encoder (MAE)



Masked Auto-Encoder (MAE)

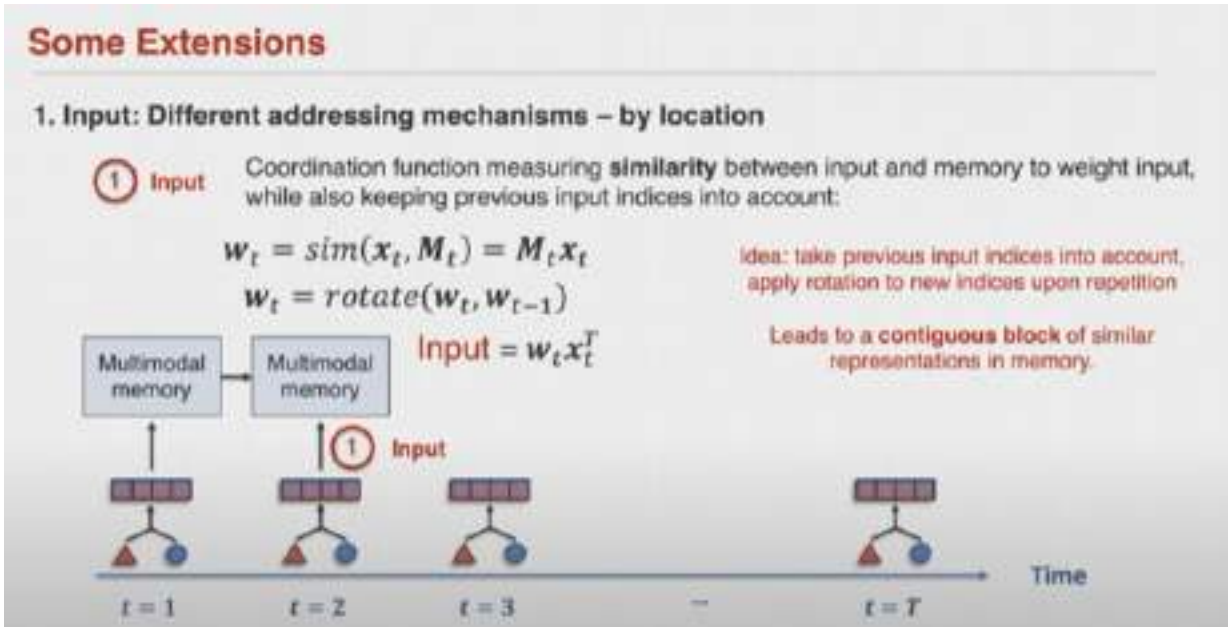


Visual-and-Language Transformer (ViLT) (≈ BERT + ViT)



Lecture 11 :MultiModal Reasoning

In this lecture we are trying to do multi modal with time by using memory.We use LSTMs ,transformers etc to use .



We need to keep memory short as it cannot be infinite and therefore we need to use an erase function as shown below:

Some Extensions

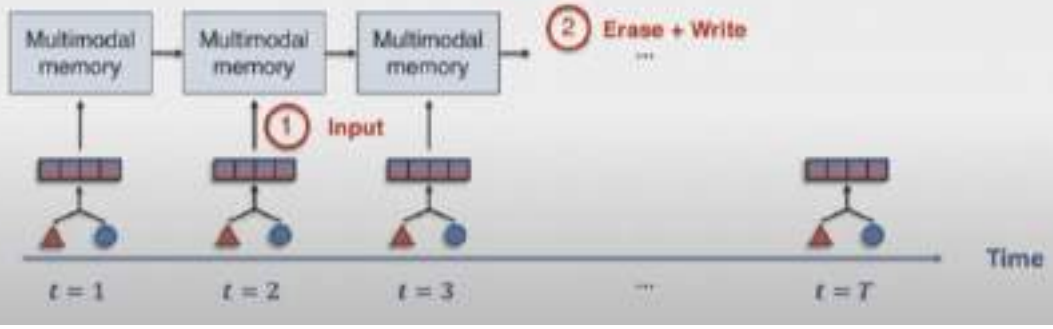
2. Writing: Including both erase and write functions

(2) Erase + write

$$M_t = M_t[1 - \alpha_t \text{Erase}]$$

$$M_{t+1} = M_t + \alpha_t \text{Input}$$

Erase: learnable vector of [0,1]



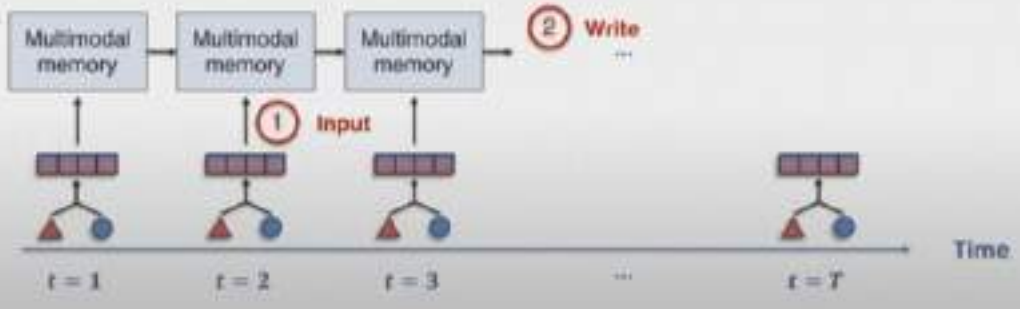
Some Extensions

3. More exponential moving average

(2) Write **Write** new addition into memory

$$M_{t+1} = (1 - \alpha_t) M_t + \alpha_t \text{Input}$$

Exponential moving average function
- Smooth out short-term fluctuations
- Highlight long-term trends



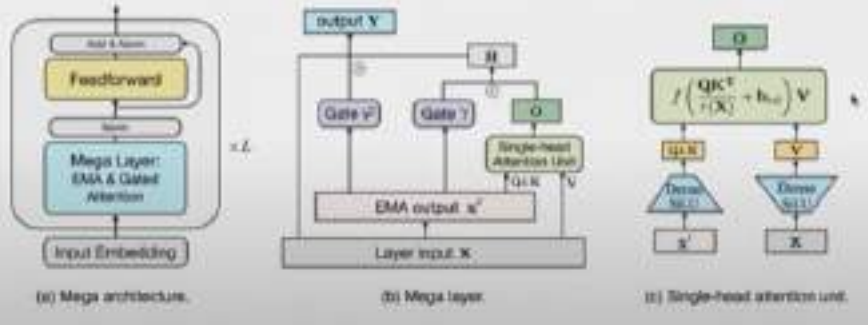
Some Extensions

3. More exponential moving average + combine with Transformers

(2) Write **Write** new addition into memory

$$M_{t+1} = (1 - \alpha_t)M_t + \alpha_t \text{Input}$$

Exponential moving average function
- Smooth out short-term fluctuations
- Highlight long-term trends



(a) Mega architecture. (b) Mega layer. (c) Single-head attention unit.

Some Extensions

4. From recurrent to parallel convolutions

A lot of what we presented seemed to be recurrent, which may not seem easily parallelizable.

But many of these have equivalent formulations in convolutional representations.

Key idea: exponential moving average can be implemented as convolution.

(2) Write **Write** new addition into memory

$$M_{t+1} = (1 - \alpha_t)M_t + \alpha_t \text{Input} \qquad M_T = K * [\text{Input\_1}, \ldots \text{Input\_T}]$$

BUT: K will be huge, the size of the entire sequence.

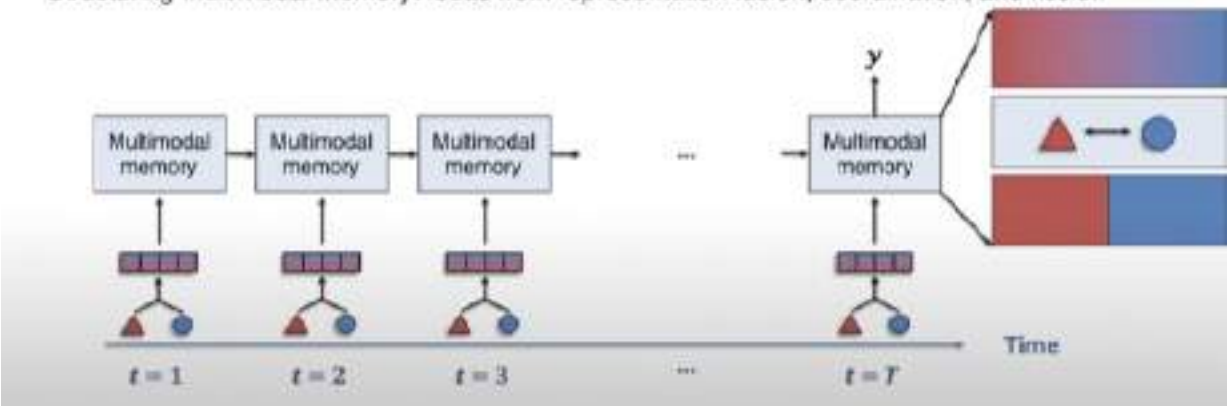
Many approximations, optimizations, see references below.

Inside the memory ,every element can be represented as fused ,coordinated or factorised representation

Memory for Multimodal Sequences

1. Memory + representation

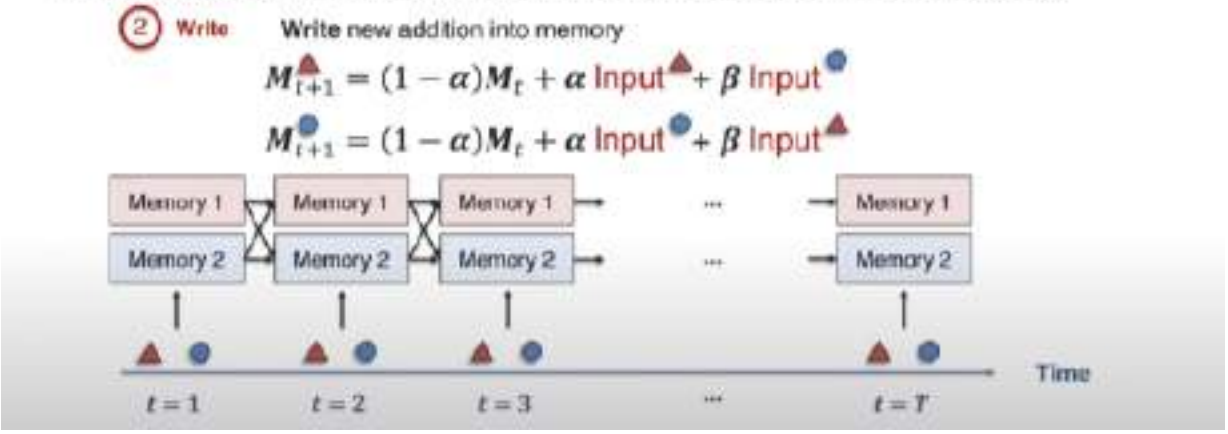
We've seen early fusion of raw modalities
Structuring multimodal memory: ideas from representation fusion, coordination, and fission



Memory for Multimodal Sequences

1. Memory + representation

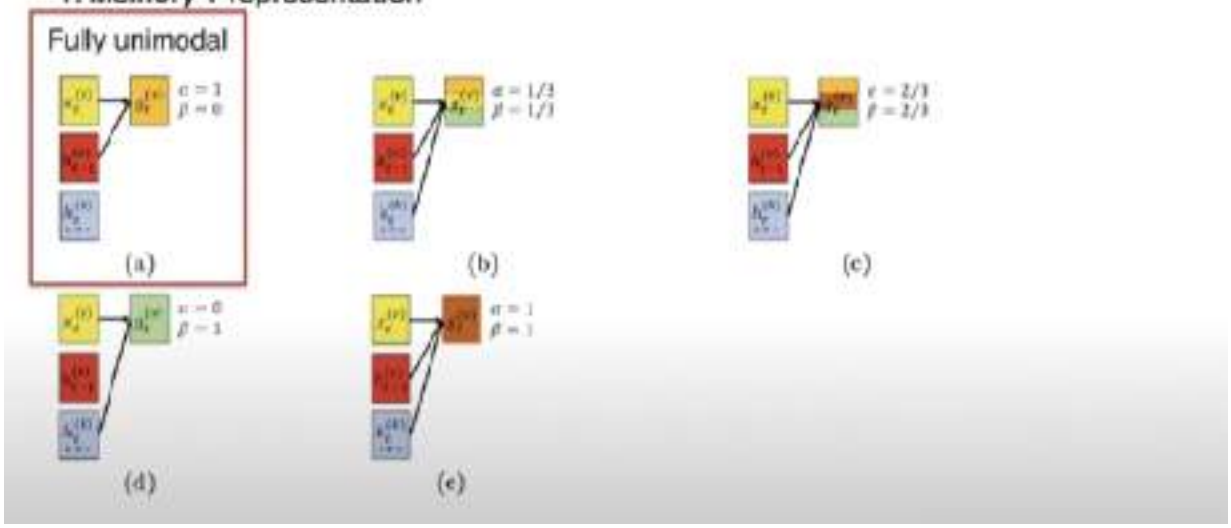
Representation can be learned not just prior to memory but also inside memory cells



We have different combinations as shown below:

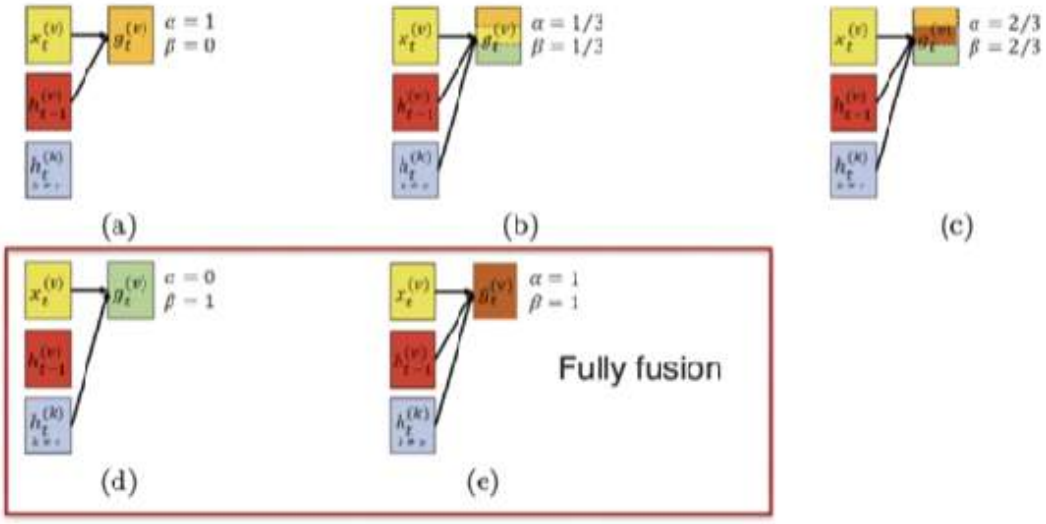
Memory for Multimodal Sequences

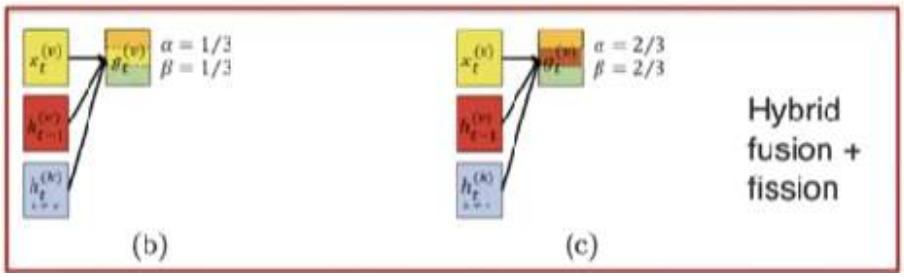
1. Memory + representation



Memory for Multimodal Sequences

1. Memory + representation

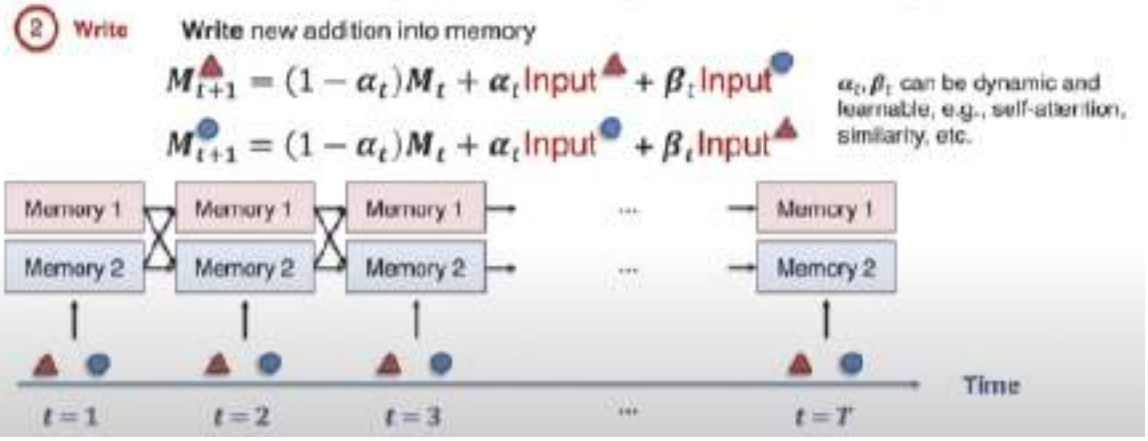




Memory for Multimodal Sequences

1. Memory + representation

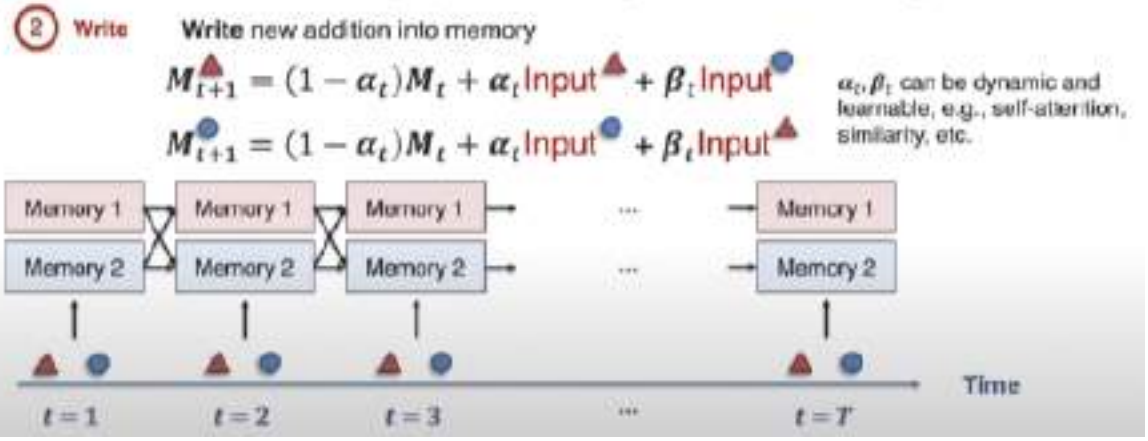
Representation can be learned not just prior to memory but also inside memory cells



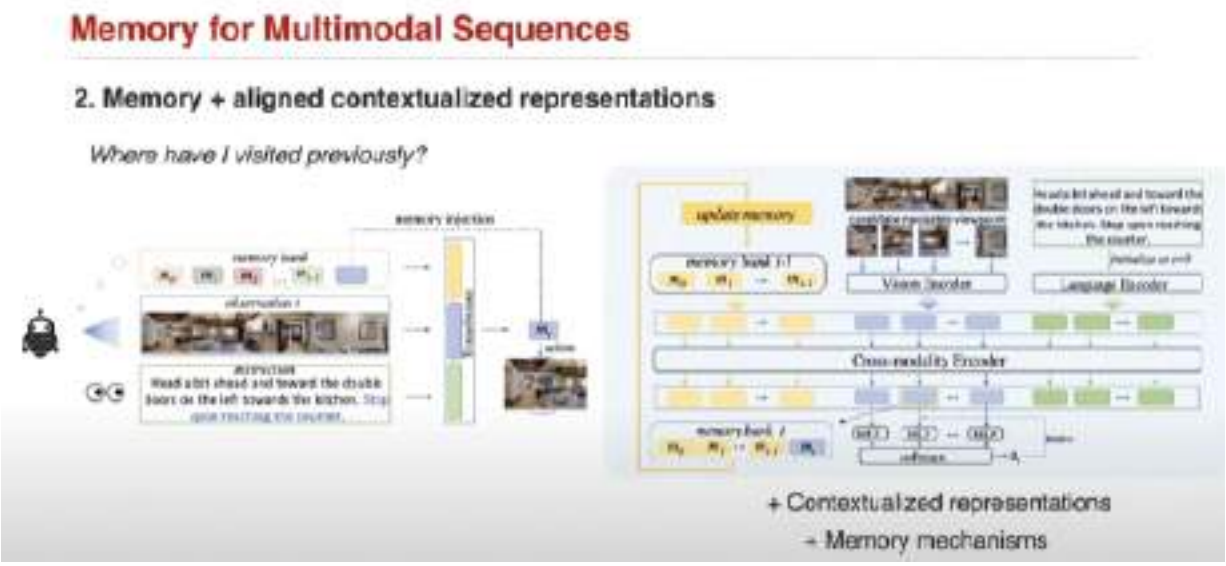
Memory for Multimodal Sequences

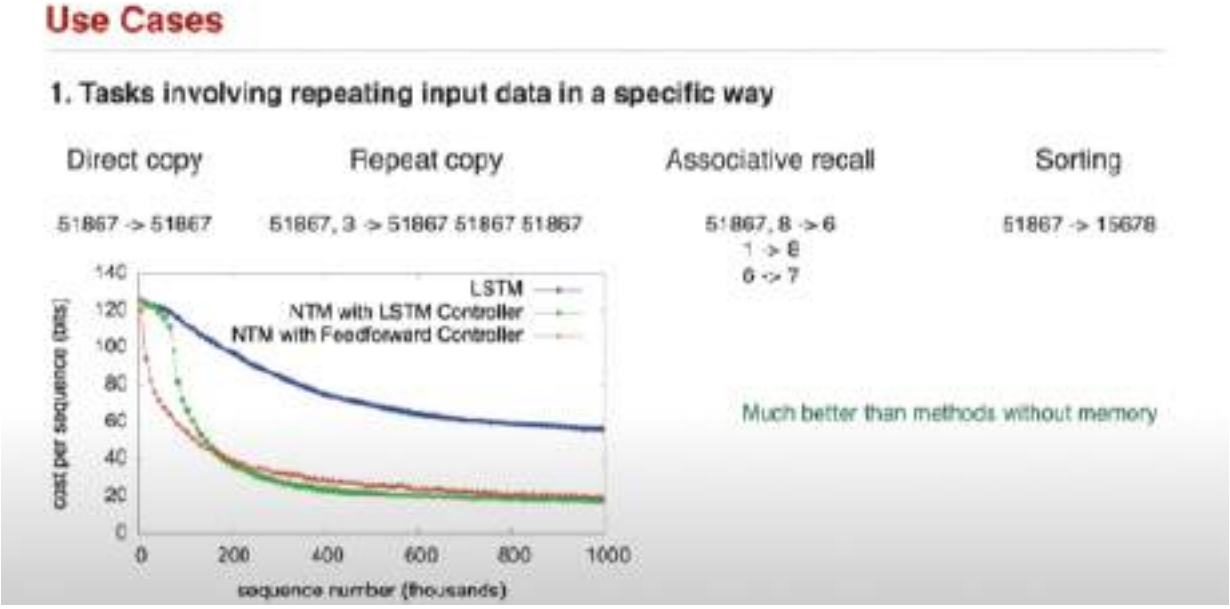
1. Memory + representation

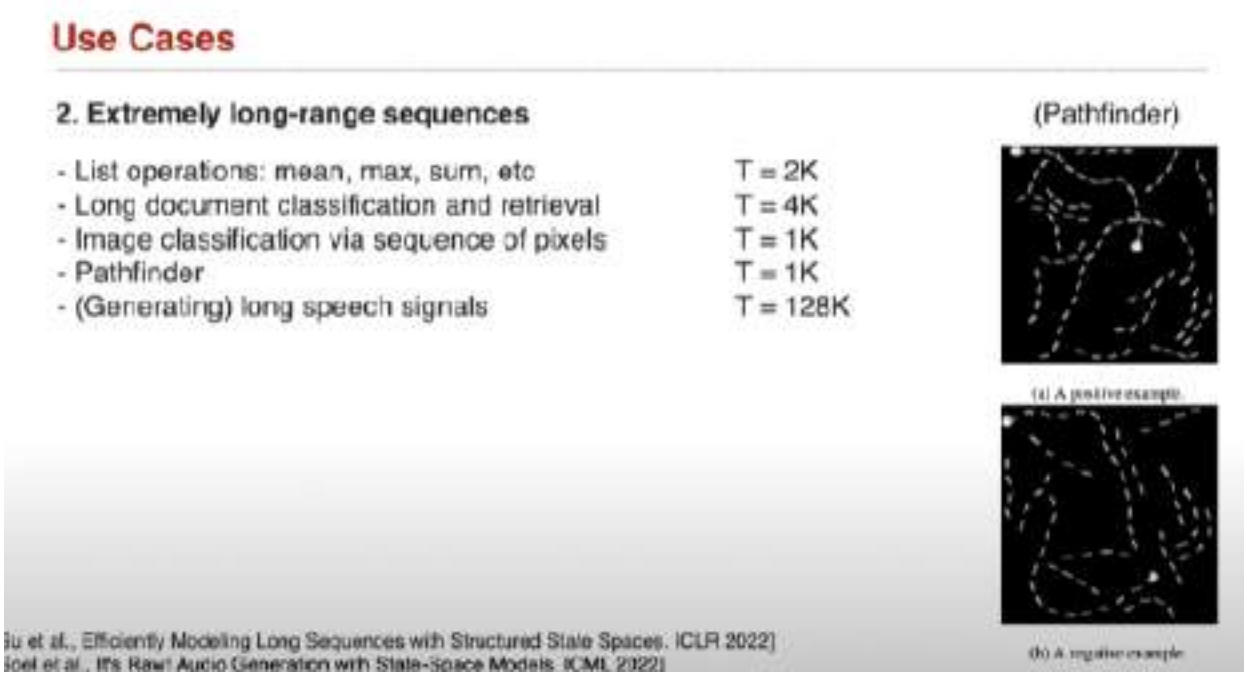
Representation can be learned not just prior to memory but also inside memory cells

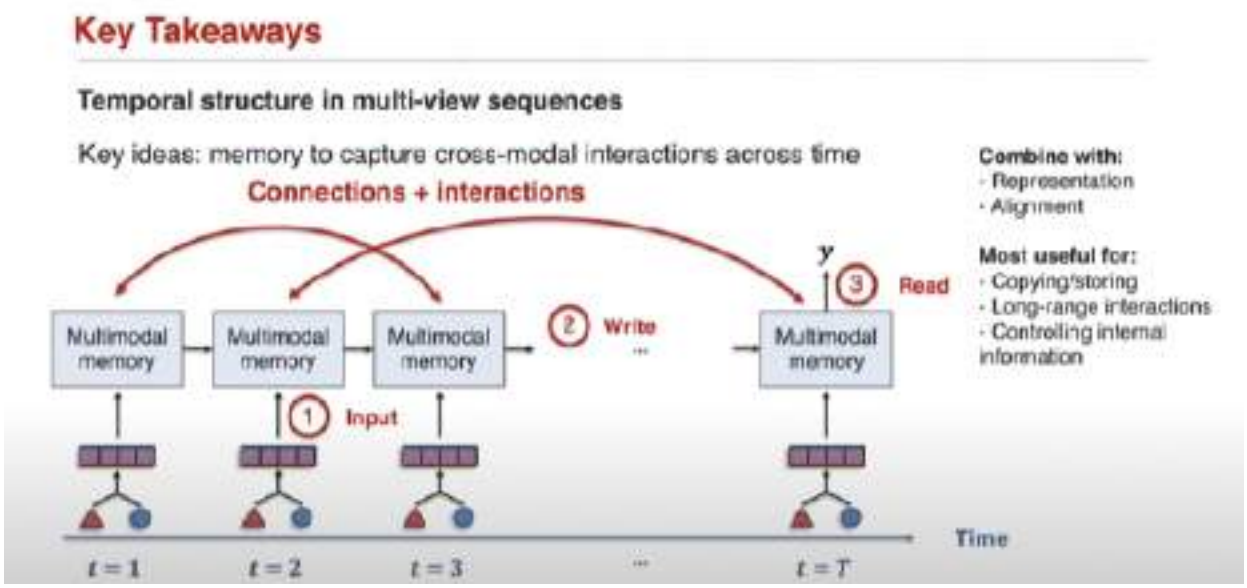


We can take an example of a robot which takes in two modalities ,stores them in memory and uses transformers to make decision:

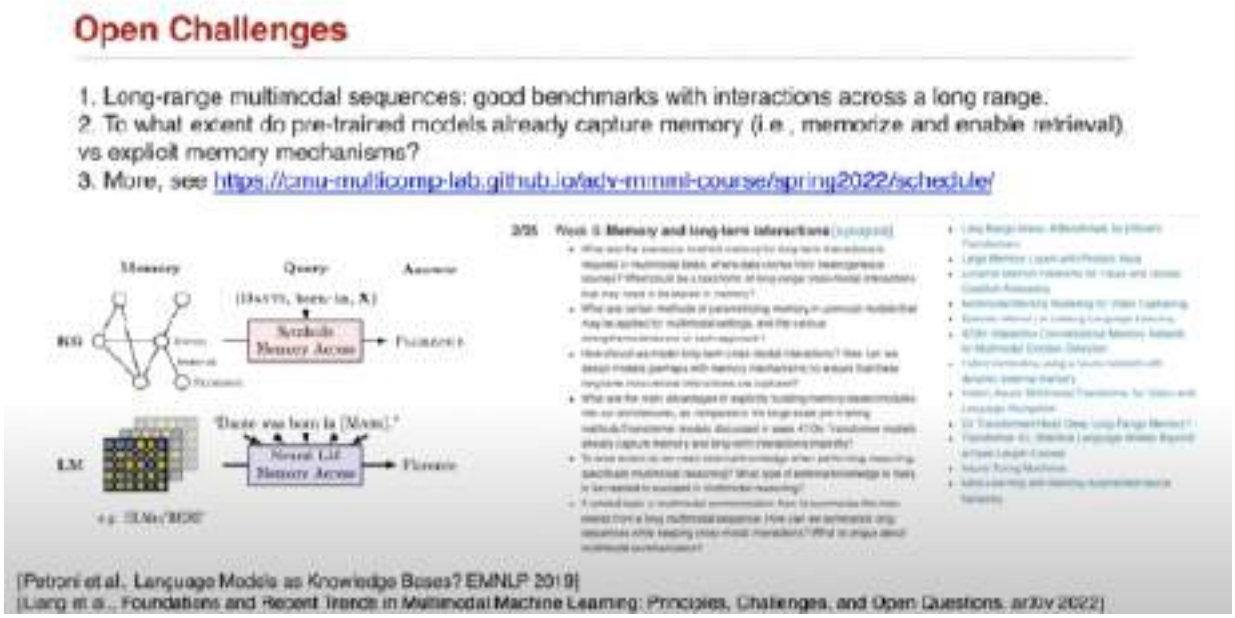








We have these kinds of challenges:

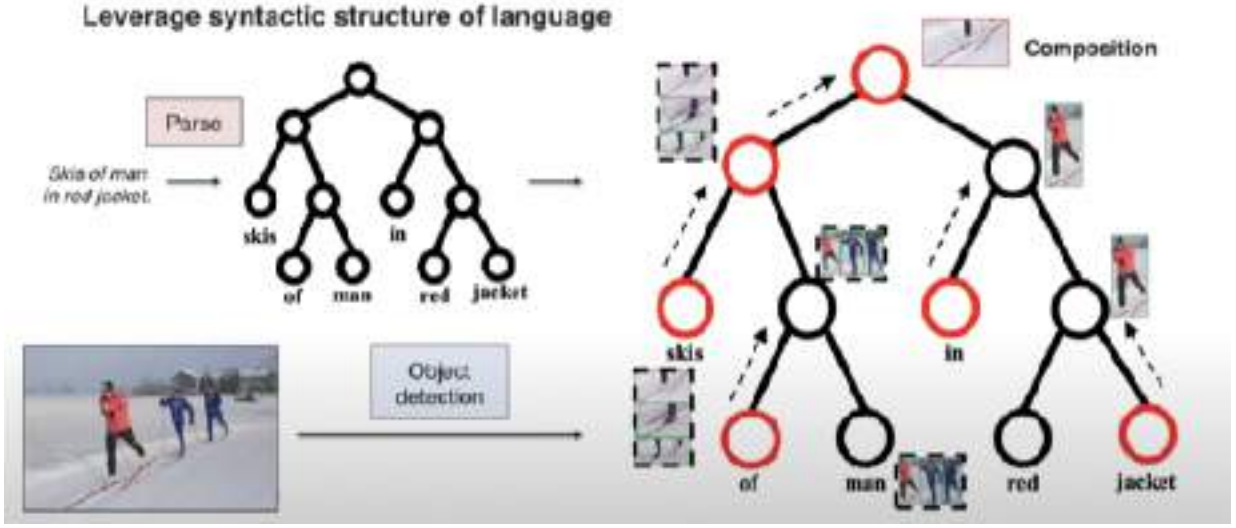


Hierarchical Representation:

In this method we get to see that the sentences which are parsed in the form shown below and then coordination representation is used to see how close they are to the objects in the pictures

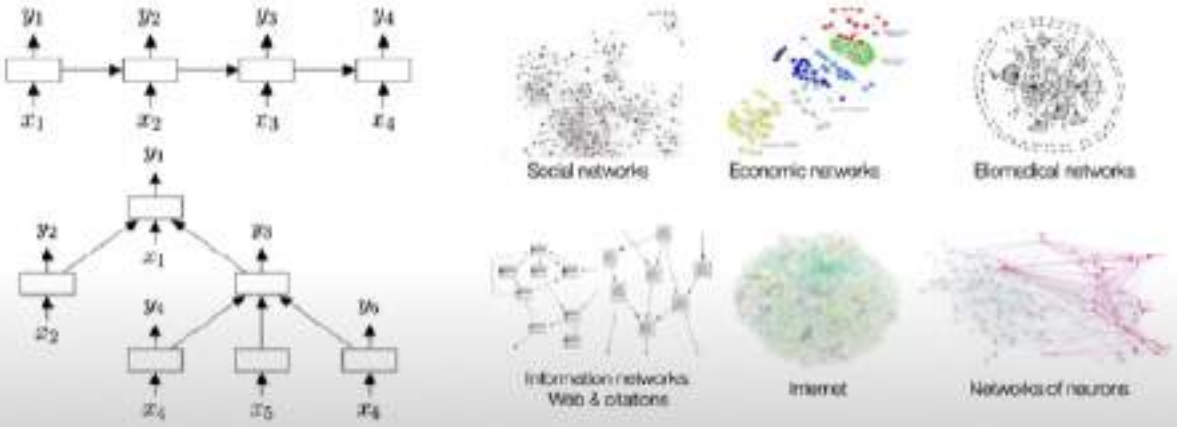
Hierarchical Structure

Leverage syntactic structure of language



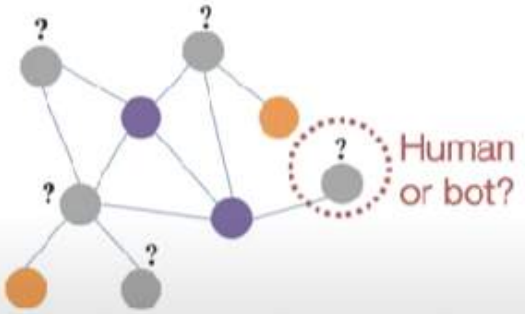
Tree and Graph Networks

From linear chain models to tree and graph-structured models



Graphs – Supervised Task

Goal: Learn from labels associated with a subset of nodes (or with all nodes)



e.g., an online social network

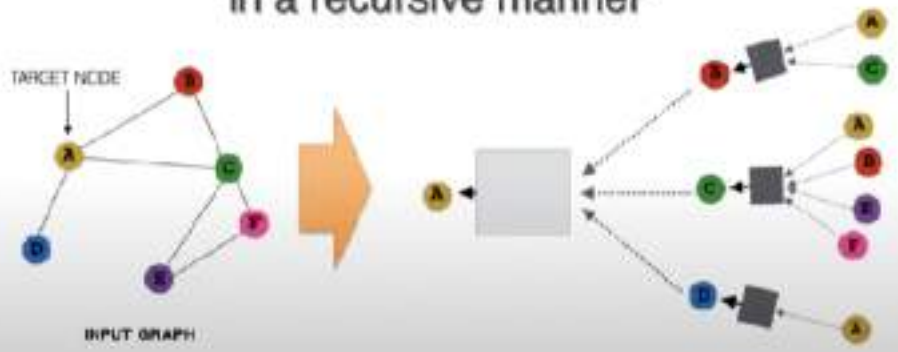
Graph Neural Nets

Assume we have a graph **G**:

- **V** is the set of vertices
- **A** is the binary adjacency matrix
- **X** is a matrix of node features:
  - Categorical attributes, text, image data
    e.g. profile information in a social network
  - ...
- **Y** is a vector of node labels (optional)

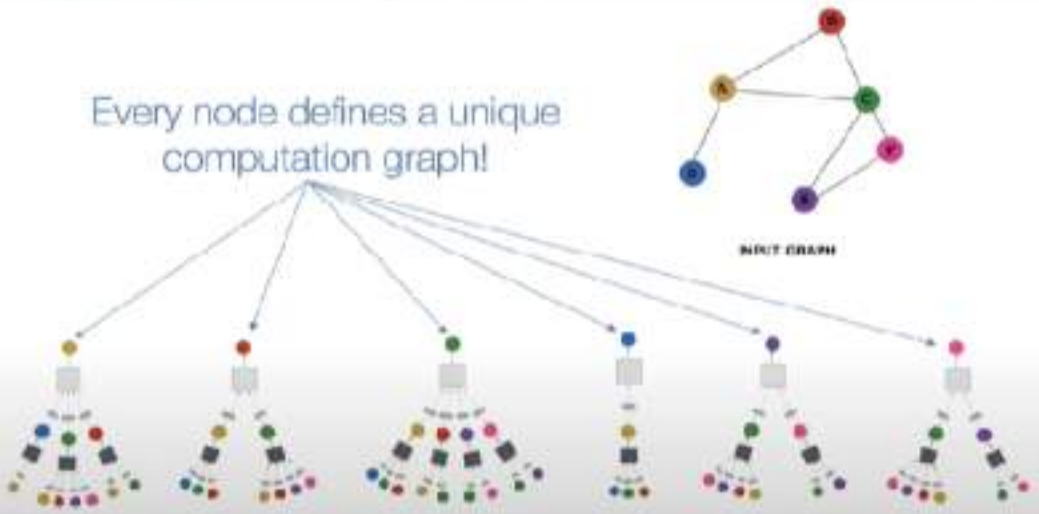
Graph Neural Nets

Key idea: Generate node embeddings based on local neighborhoods in a recursive manner



Graph Neural Nets

Every node defines a unique computation graph!

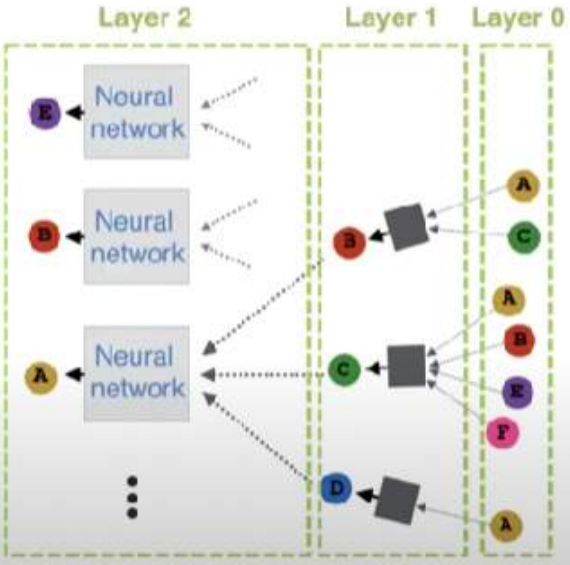


Graph Neural Nets

And multiple layers!

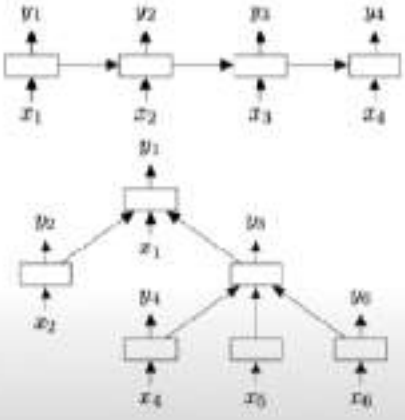
- Shared parameters within a specific layer
- "layer-0" is the input feature x_v

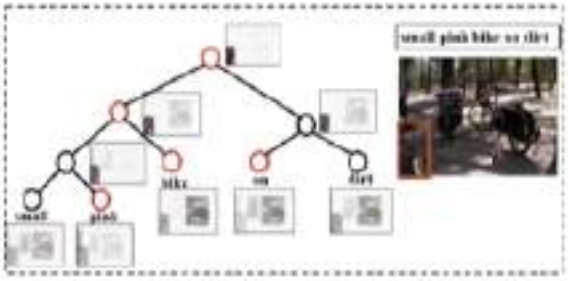
INPUT GRAPH



Experiments

From linear chain models to tree models





RefCOCO

Accounting for syntactic structure also improves language-based sentiment analysis, semantic matching, question-answering, language modeling, interpreting attention scores, etc.

How do Graph Nets Work?

Empirically graph nets work well over less structured networks, but why?

Key idea: algorithmic alignment - link compositional structure required for task with computational structure of prediction model

Summary statistics
What is the maximum value difference among treasures?

Relational argmax
What are the colors of the farthest pair of objects?

Dynamic programming
What is the cost to defeat monster X by following the optimal path?

NP-hard problem
Subset sum: Is there a subset that sums to 0?

How do Graph Nets Work?

Empirically graph nets work well over less structured networks, but why?

Key idea: algorithmic alignment - link compositional structure required for task with computational structure of prediction model

Many multimodal reasoning problems here:
intuitive physics, visual question answering, shortest paths

Summary statistics
What is the maximum value difference among treasures?

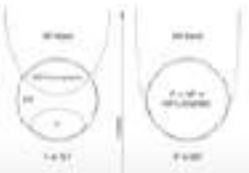
DeepSets

Relational argmax
What are the colors of the farthest pair of objects?

1-layer GNN

Dynamic programming
What is the cost to defeat monster X by following the optimal path?

K-layer GNN



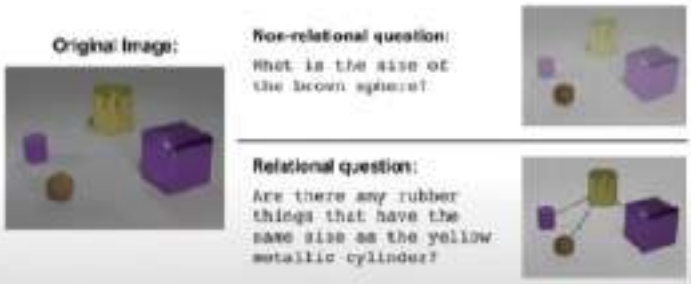
NP-hard problem
Subset sum: Is there a subset that sums to 0?

None ☹

How do Graph Nets Work?

Empirically: datasets that require multiple steps of relational reasoning

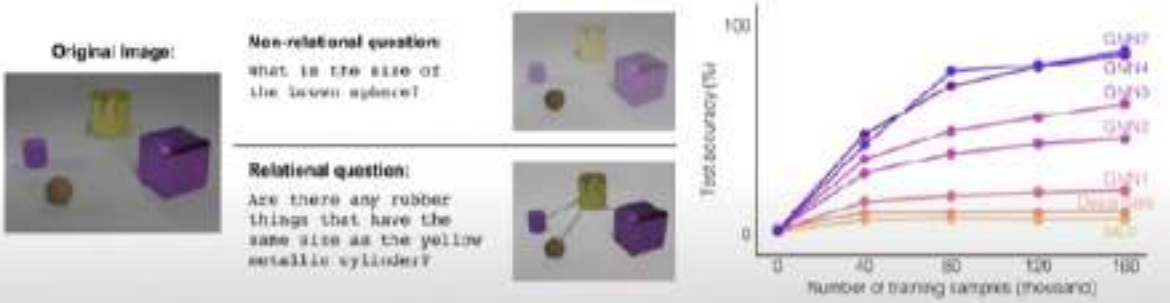
1. Sudoku: number interactions, multi-step, backtracking,
2. Relational VQA: CLEVR -> Sort-of-CLEVR -> Pretty-CLEVR (*'which object is closest/k-steps away'*)



How do Graph Nets Work?

Empirically: datasets that require multiple steps of relational reasoning

1. Sudoku: number interactions, multi-step, backtracking,
2. Relational VQA. CLEVR -> Sort-of-CLEVR -> Pretty-CLEVR (*'which object is closest/k-steps away'*)



Key Takeaways & Open Challenges

1. Relations are between elements from same modality, so distances and representations are well-defined.
   -> how to handle cross-modal interconnections at the same time?
2. Heterogeneous graph nets, where nodes come from different modalities.
3. Formal connections between cross-modal interactions and relational reasoning.
4. Quantifying the reasoning required by decomposing datasets into perception vs reasoning.

Summary statistics
What is the maximum value difference among treasures?

Relational argmax
What are the colors of the furthest pair of objects?

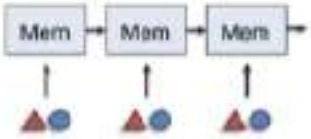
Dynamic programming
What is the cost to defeat monster X by following the optimal path?

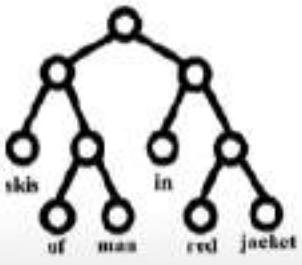
NP-hard problem
Subset sum: Is there a subset that sums to 0?

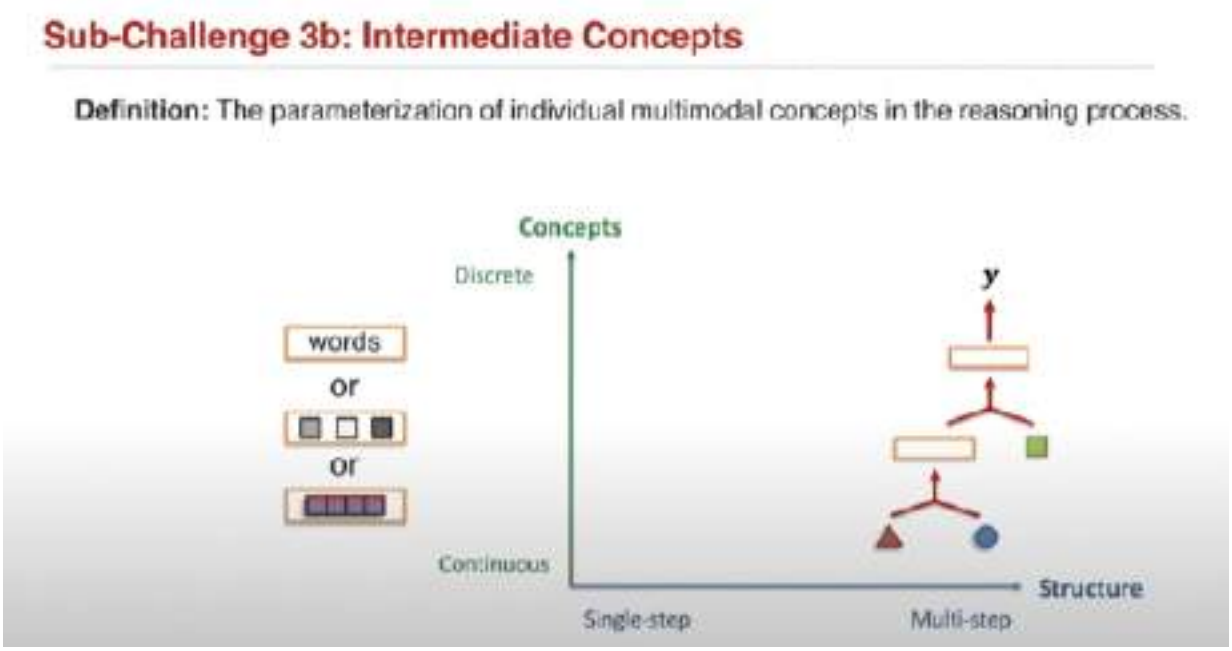
Summary

Reasoning is about compositionality, and compositionality requires knowing the structure.

In the continuous case (i.e., if structure is given or can be learned easily in a differentiable manner):



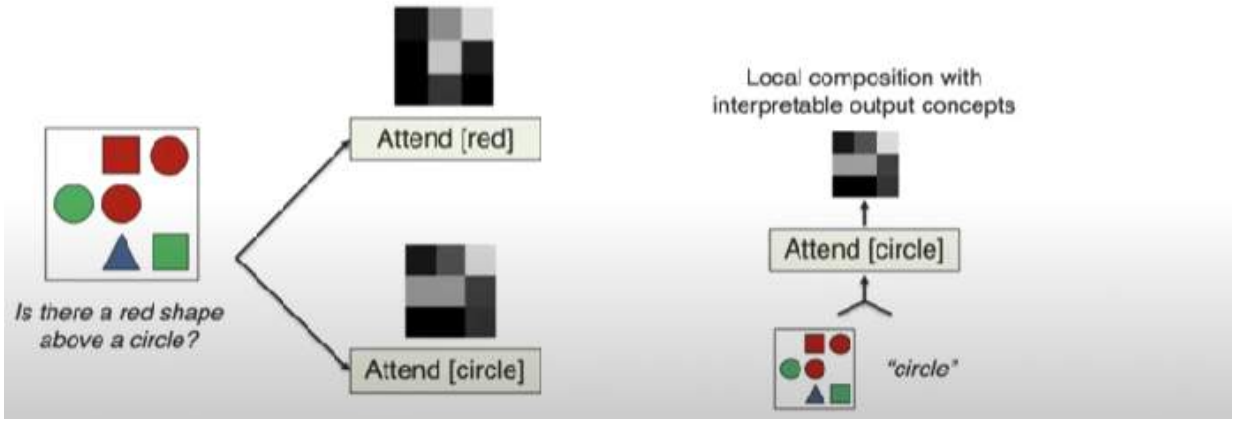




In the concept below we can create the following attention maps by using a transformers and CNNs as shown below:

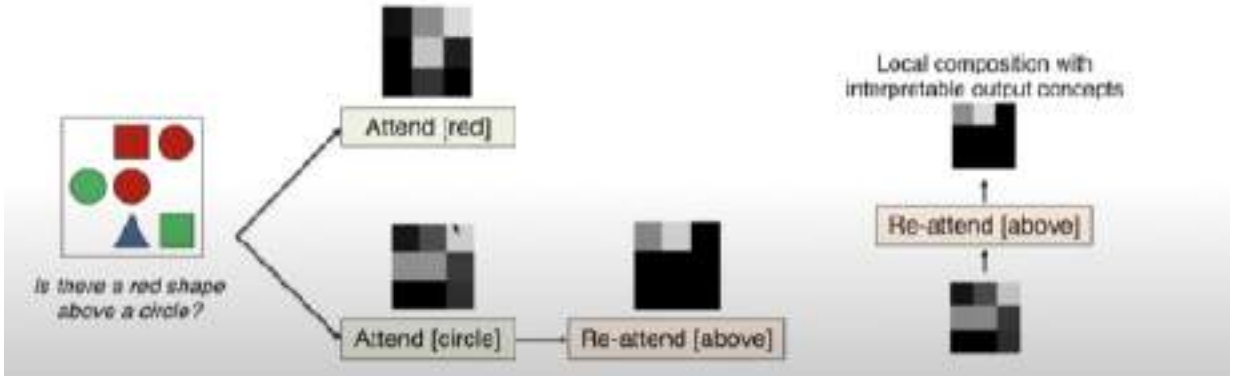
Neuro-symbolic Concepts

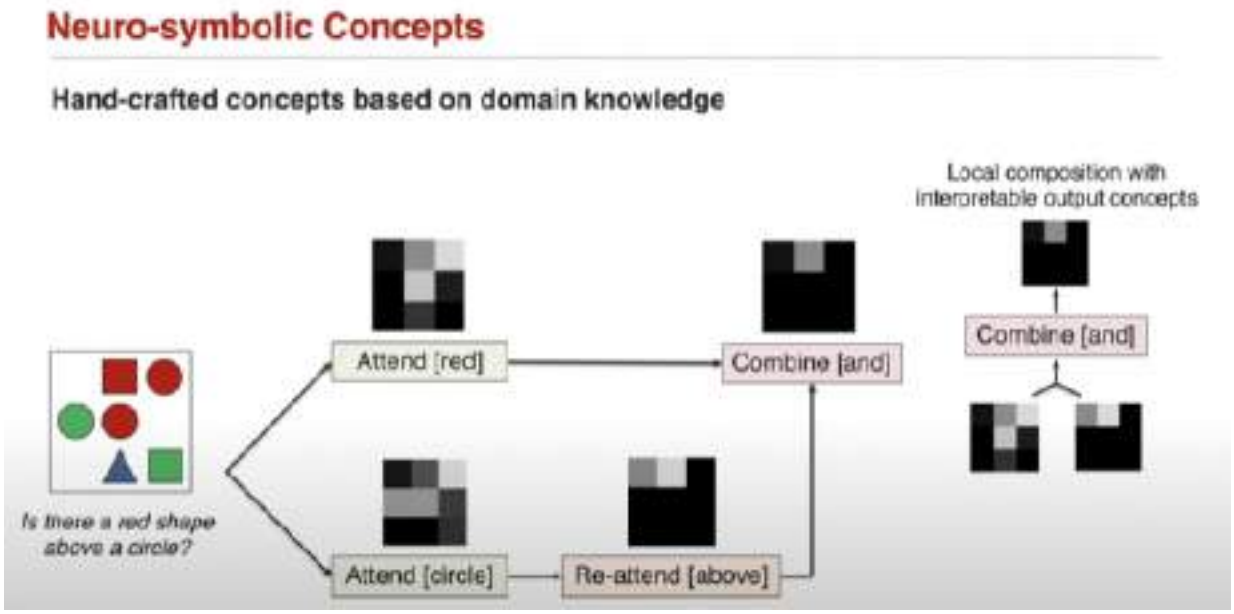
Hand-crafted concepts based on domain knowledge

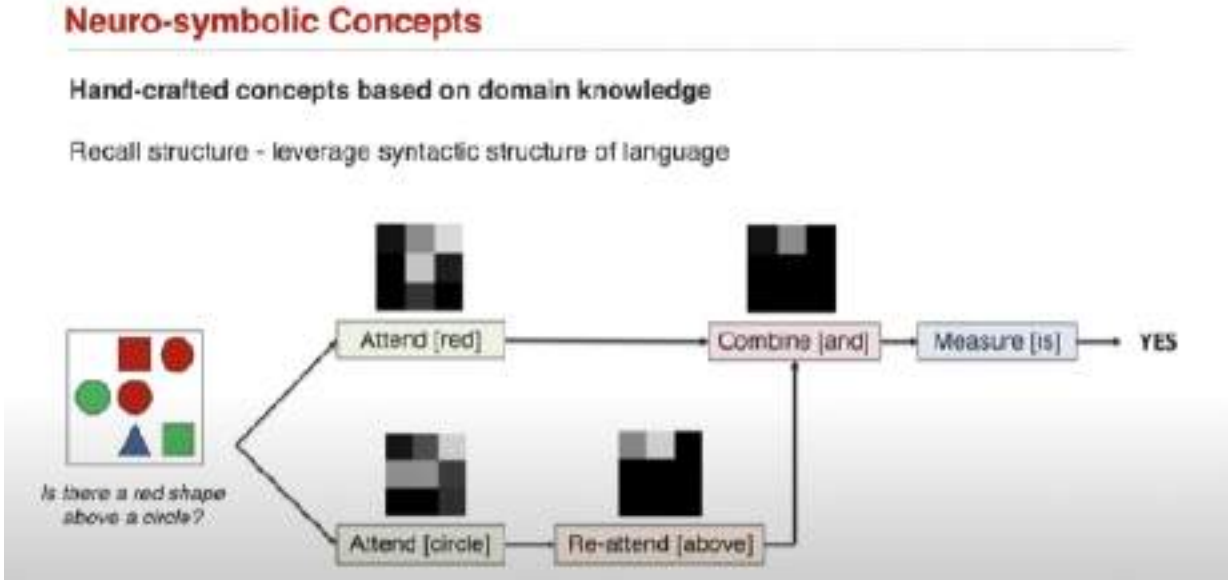


Neuro-symbolic Concepts

Hand-crafted concepts based on domain knowledge

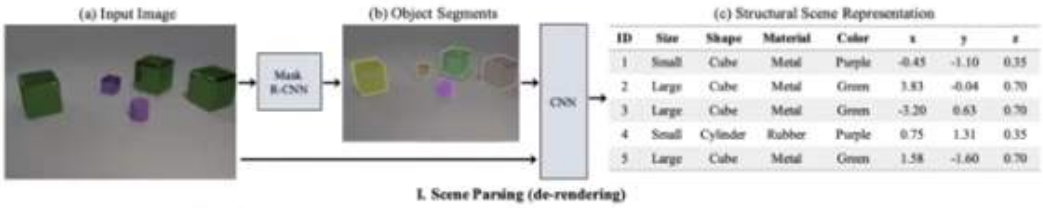






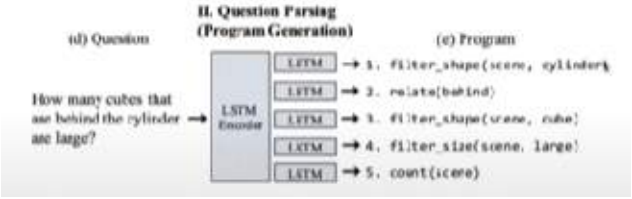
In the above we take a question parse into small parts ,apply operations such as creating attention maps using transformers and then we combine them as shown above
We can take an image and parse it in the same manner and give each detected objects some properties as shown in the image below:

Hand-crafted concepts based on domain knowledge



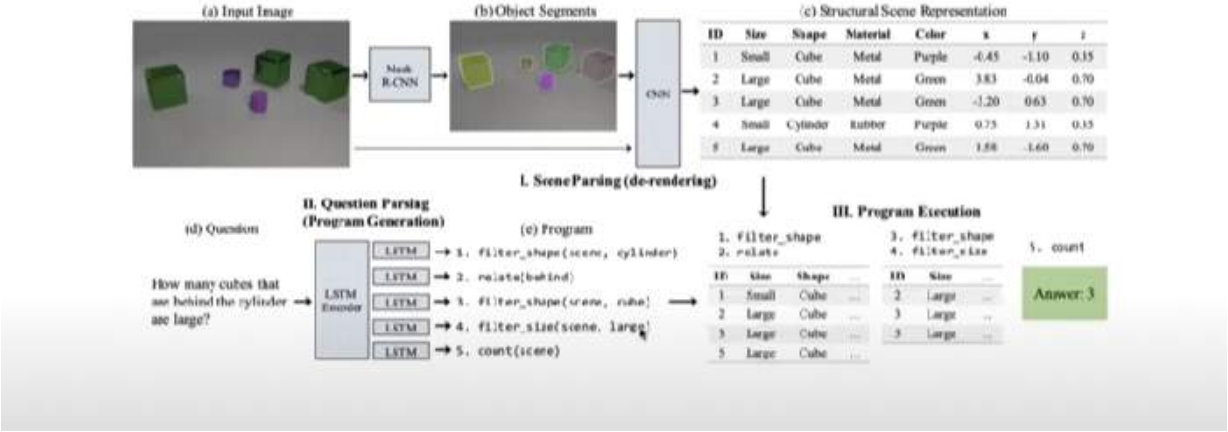
ID	Size	Shape	Material	Color	x	y	z
1	Small	Cube	Metal	Purple	-0.45	-1.10	0.35
2	Large	Cube	Metal	Green	3.83	-0.04	0.70
3	Large	Cube	Metal	Green	-3.20	0.63	0.70
4	Small	Cylinder	Rubber	Purple	0.75	1.31	0.35
5	Large	Cube	Metal	Green	1.58	-1.60	0.70

Than we can ask a question and apply the same method as shown below :



More Neuro-symbolic Concepts

Hand-crafted concepts based on domain knowledge



ID	Size	Shape	Material	Color	x	y	z
1	Small	Cube	Metal	Purple	-0.45	-1.10	0.35
2	Large	Cube	Metal	Green	3.83	-0.04	0.70
3	Large	Cube	Metal	Green	-3.20	0.63	0.70
4	Small	Cylinder	Rubber	Purple	0.75	1.31	0.35
5	Large	Cube	Metal	Green	1.58	-1.60	0.70

Summary: Reasoning Part 1

Definition: Combining knowledge, usually through multiple inferential steps, exploiting multimodal alignment and problem structure.

(a) some plants surrounding a lightbulb

(b) a lightbulb surrounding some plants

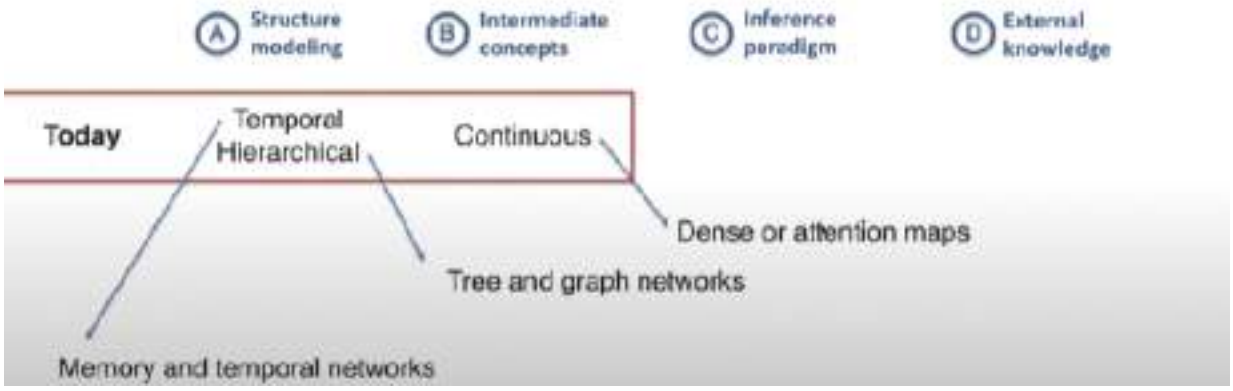
CLIP, ViLT, VILBERT, etc.
All random chance

Compositional Generalization
to novel combinations outside
of training data

1. Structure: <subject> <verb> <object>
2. Concepts: 'plants', 'lightbulb'
3. Inference: 'surrounding' – spatial relation
4. Knowledge: from humans!

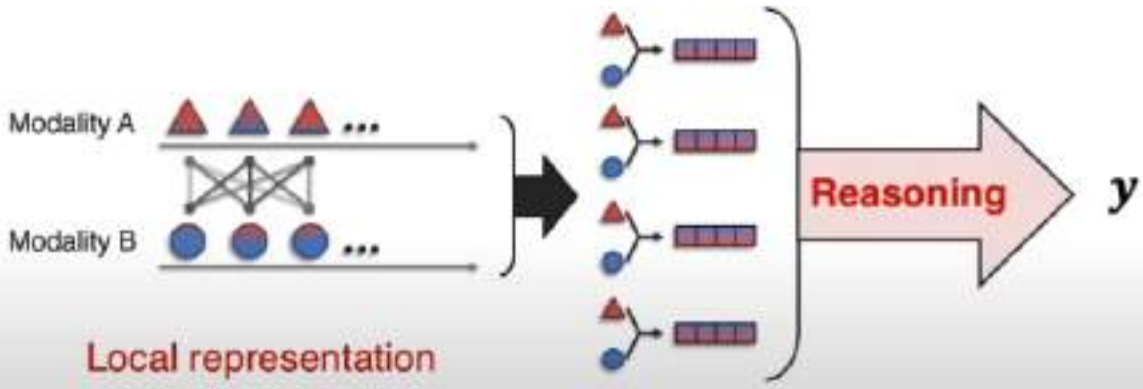
Summary: Reasoning Part 1

Definition: Combining knowledge, usually through multiple inferential steps, exploiting multimodal alignment and problem structure.



Reasoning

Definition: Combining knowledge, usually through multiple inferential steps, exploiting multimodal alignment and problem structure.



Local representation
+ Aligned representation

The Challenge of Compositionality

Definition: Combining knowledge, usually through multiple inferential steps, exploiting multimodal alignment and problem structure.

(a) some plants surrounding a lightbulb

(b) a lightbulb surrounding some plants

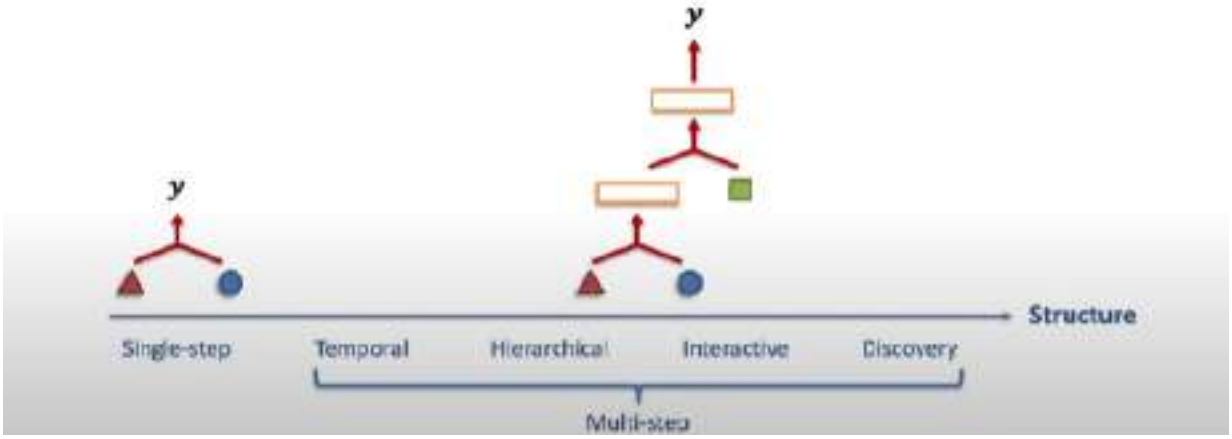
CLIP, ViLT, ViLBERT, etc.
All random chance

Compositional Generalization
to novel combinations outside of training data

1. Structure: <subject> <verb> <object>
2. Concepts: 'plants', 'lightbulb'
3. Inference: 'surrounding' – spatial relation
4. Knowledge: from humans!

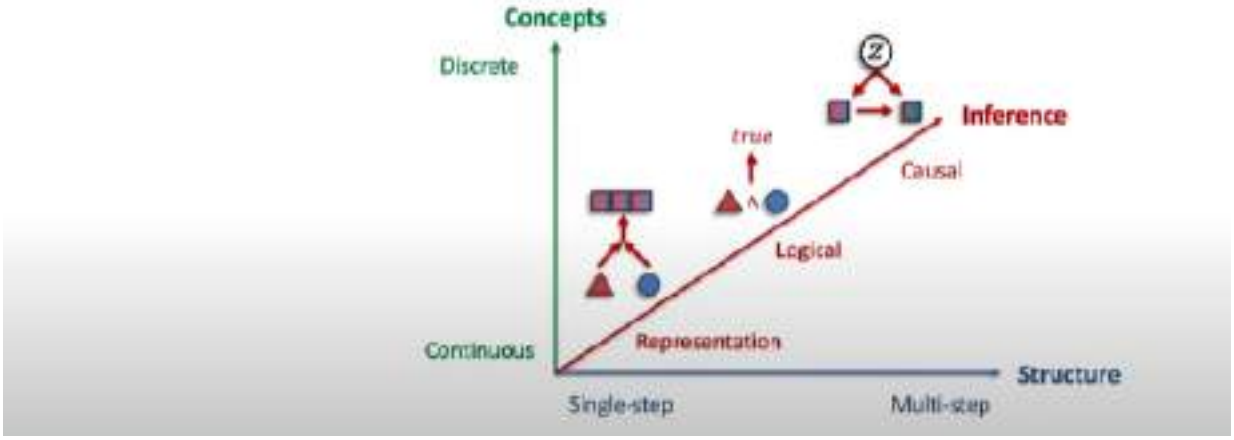
Sub-Challenge 3a: Structure Modeling

Definition: Defining or learning the relationships over which reasoning occurs.



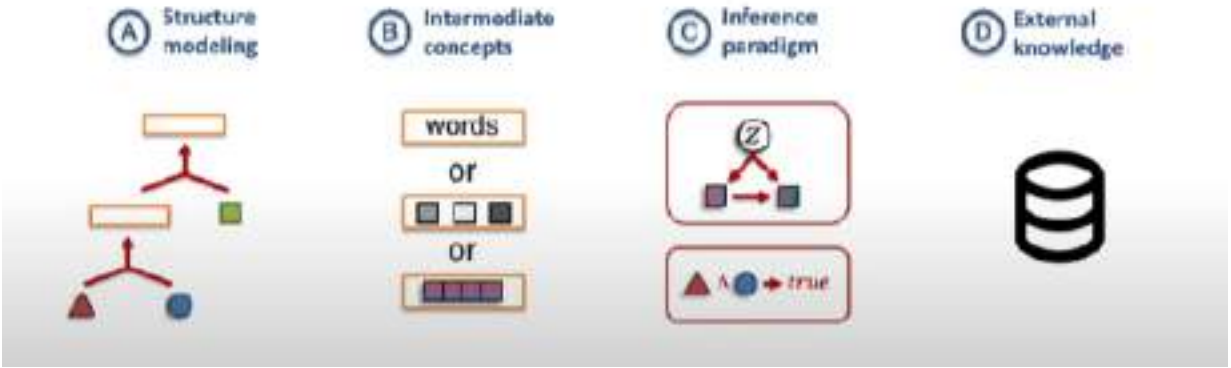
Sub-Challenge 3c: Inference Paradigm

Definition: How increasingly abstract concepts are inferred from individual multimodal evidences.



Reasoning

Definition: Combining knowledge, usually through multiple inferential steps, exploiting multimodal alignment and problem structure.

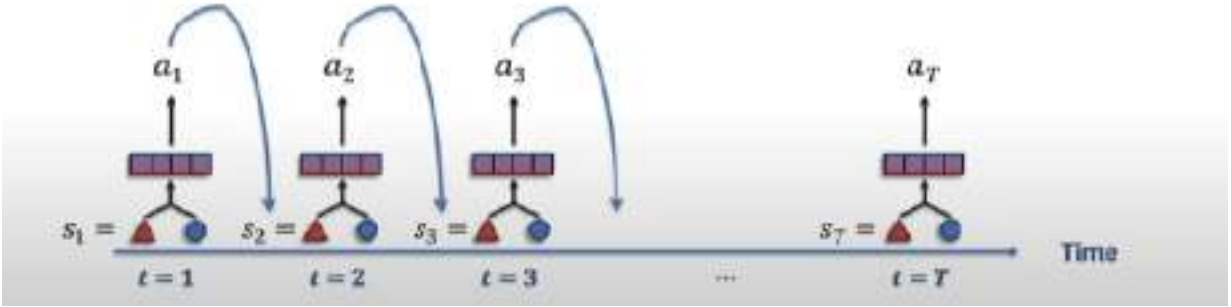


Interactive Structure

Structure defined through interactive environment

Main difference from temporal - actions taken at previous time steps affect future states

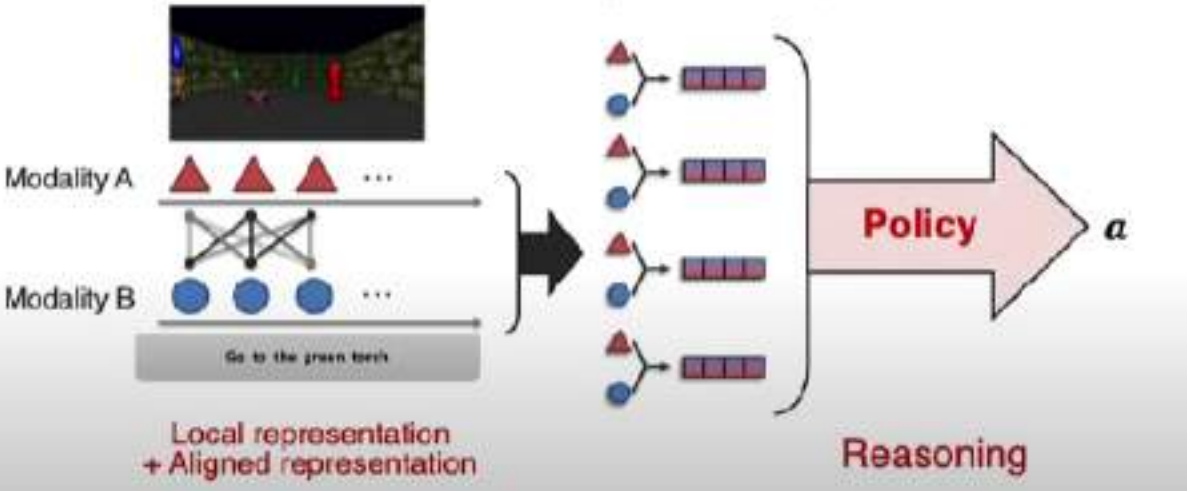
Integrates multimodality into the reinforcement learning framework

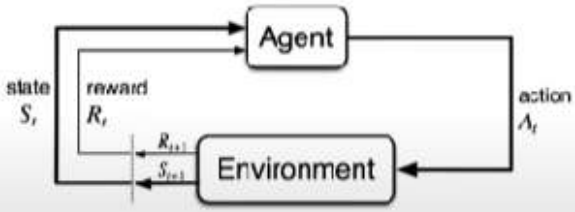


Interactive Structure

Structure defined through interactive environment

Main difference from temporal - actions taken at previous time steps affect future states

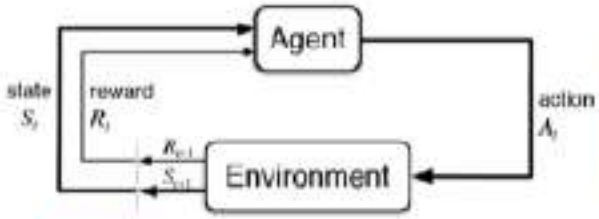




Learning a Policy – RL basics

An MDP is defined by:

- Set of states S.
- Set of actions A.
- Transition function $P(s'|s,a)$.
- Reward function $r(s,a,s')$.
- Start state s_0.
- Discount factor γ.
- Horizon H.



Return:

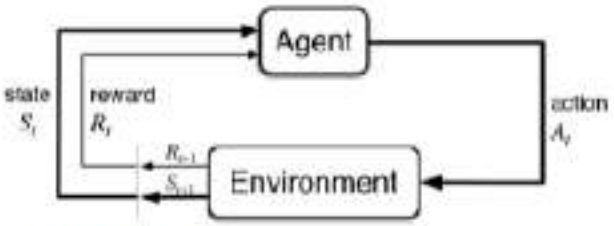
$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Policy: $\pi(a|s) = \Pr(A_t = a|S_t = s) \quad \forall t$

Learning a Policy – RL basics

An MDP is defined by:

- Set of states S.
- Set of actions A.
- Transition function $P(s'|s,a)$.
- Reward function $r(s,a,s')$.
- Start state s_0.
- Discount factor γ.
- Horizon H.



Return:

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

π:

Policy: $\pi(a|s) = \Pr(A_t = a|S_t = s) \quad \forall t$

Goal: $\arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{H} \gamma^t R_t|\pi\right]$

RL vs Supervised Learning

Reinforcement Learning

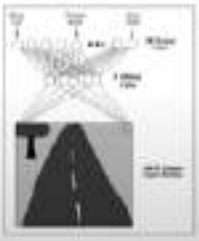
- Sequential decision making
- Maximize cumulative reward
- Sparse rewards
- Environment maybe unknown

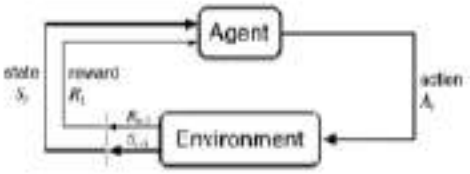
Supervised Learning

- One-step decision making
- Maximize immediate reward
- Dense supervision
- Environment always known

Intersection Between RL and Supervised Learning

Imitation learning





Obtain expert trajectories (e.g. human driver/video demonstrations):

$$s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \ldots$$

Perform supervised learning by predicting expert action

$$D = \{(s_0, a_0^*), (s_1, a_1^*), (s_2, a_2^*), \ldots\}$$

But: distribution mismatch between training and testing
Hard to recover from sub-optimal states
Sometimes not safe/possible to collect expert trajectories

Optimal State and Action Value Functions

Definitions

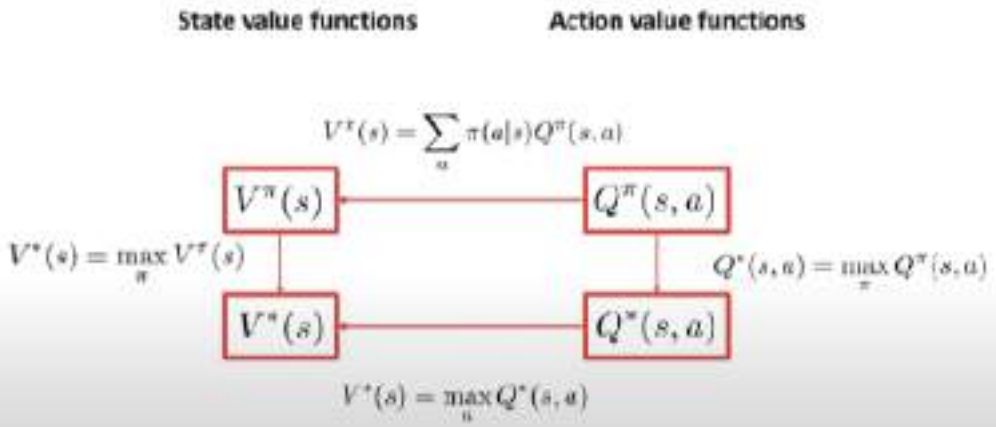
- Definition: the **optimal state-value function** $V^*(s)$ is the maximum value function over all policies

$$V^*(s) = \max_\pi V^\pi(s)$$

- Definition: the **optimal action-value function** $Q^*(s,a)$ is the maximum action-value function over all policies

$$Q^*(s,a) = \max_\pi Q^\pi(s,a)$$

Relationships Between State and Action Values



Obtaining the Optimal Policy

Optimal policy can be found by maximizing over Q*(s,a)

$$\pi^*(a|s) = \begin{cases} 1, & \text{if } a = \arg\max_a Q^*(s,a) \\ 0, & \text{else} \end{cases}$$

Optimal policy can also be found by maximizing over V*(s') with **one-step look ahead**

$$\pi^*(a|s) = \begin{cases} 1, & \text{if } a = \arg\max_a \mathbb{E}_{s'}\left[r(s,a,s') + \gamma V^*(s')\right] \\ 0, & \text{else} \end{cases}$$

$$\pi^*(a|s) = \begin{cases} 1, & \text{if } a = \arg\max_a \left[\sum_{s'} p(s'|s,a)(r(s,a,s') + \gamma V^*(s'))\right] \\ 0, & \text{else} \end{cases}$$

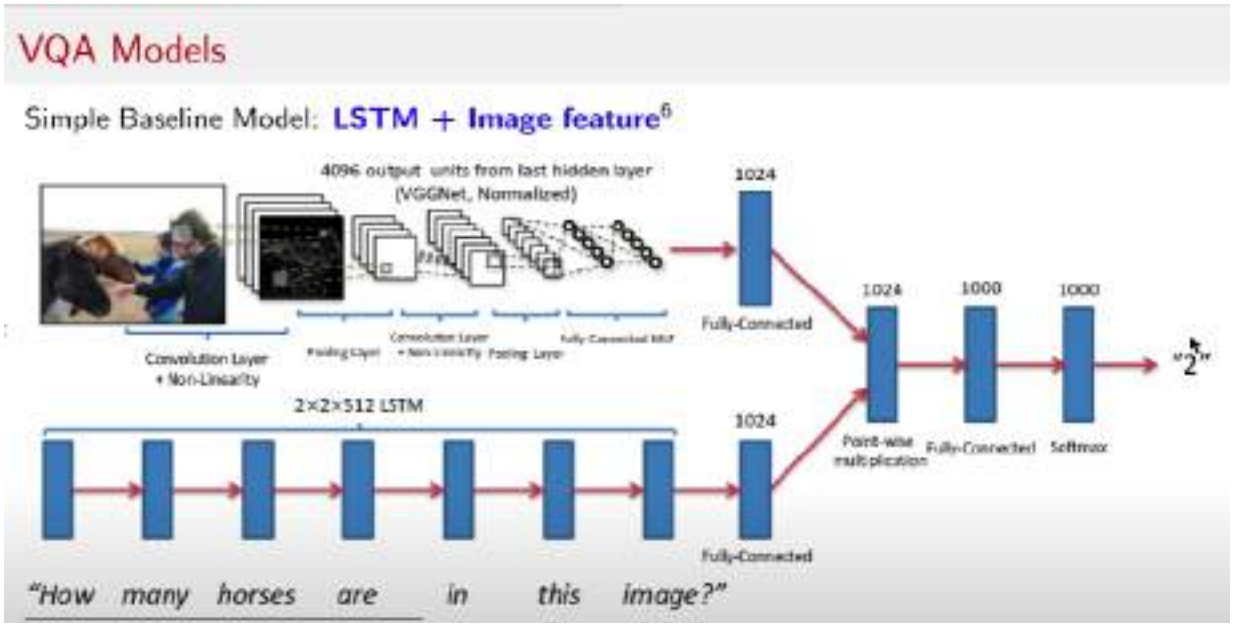
Codes:

Hugging Face VQA:https://huggingface.co/google/deplot

Simple LSTM + CNN :
https://www.researchgate.net/publication/337275202_Visual_Question_Answering_using_combination_of_LSTM_and_CNN_A_Survey

https://github.com/ntusteeian/VQA_CNN-LSTM

Visual Question Answering: Coding Notes Late Fusion



Ways in which dataset was collected:

- We collected a new dataset of “realistic” abstract scenes to enable research focused only on the high-level reasoning required for VQA by removing the need to parse real images. Three questions were collected for each image or scene. Each question was answered by ten subjects along with their confidence. The dataset contains over 760K questions with around 10M answers.
- While the use of open-ended questions offers many benefits, it is still useful to understand the types of questions that are being asked and which types various algorithms may be good at answering. To this end, we analyze the types of questions asked and the types of answers provided. Through several visualizations, we demonstrate the astonishing diversity of the questions asked. We also explore how the information content of questions and their answers differs from image captions
-

VQA Efforts. Several recent papers have begun to study visual question answering [19], [36], [50], [3]. However, unlike our work, these are fairly restricted (sometimes synthetic) settings with small datasets. For instance, [36] only considers questions whose answers come from a predefined closed world of 16 basic colors or 894 object categories. [19] also considers questions generated from templates from a fixed vocabulary of objects, attributes, relationships between objects, *etc*. In contrast, our proposed task involves *open-ended*, *free-form* questions and answers provided by humans. Our goal is to increase the diversity of knowledge and kinds of reasoning needed to provide correct answers. Critical to achieving

- In order to collect questions a user interface was created so that a variety of questions can be put into one place so that high level questions can be asked.
- we ran pilot studies asking human subjects to ask questions about a given image that they believe a "toddler", "alien", or "smart robot" would have trouble answering. We found the "smart robot" interface to elicit the most interesting and diverse questions
- When writing a question, the subjects were shown the previous questions already asked for that image to increase the question diversity

---

How was Answers collected?

- Open-ended questions result in a diverse set of possible answers. For many questions, a simple "yes" or "no" response is sufficient. However, other questions may require a short phrase. Multiple different answers may also be correct.
- we gather 10 answers for each question from unique workers, while also ensuring that the worker answering a question did not ask it
- ask the subjects to provide answers that are "a brief phrase and not a complete sentence
-

Datasets:

Coco DataSet v2 for real images:
Dataset contains :

Images :0.25 M
Questions :0.76 M
Answers: 10 M

But bad dataset is if it has :2,591 and 1,449 images respectively

Abstract Datasets:
We are using the abstract dataset which is smaller and has the following characteristics:
For abstract scenes, we created splits for standardization, separating the scenes into 20K/10K/20K for train/val/test splits, respectively.

Captions. The MS COCO dataset [32], [7] already contains five single-sentence captions for all images. We also collected five single-captions for all abstract scenes using the same user interface[1] for collection.

Questions on abstract datasets:

Image Encoder :
This image encoder has the Vgg19 model

`models.vgg19`: `models` is a module in PyTorch that provides pre-defined models for various computer vision tasks. `vgg19` refers to the VGG-19 model architecture.
`pretrained=True`: This argument specifies that the model should be loaded with pre-trained weights. VGG-19 is a deep convolutional neural network that has been trained on a large dataset, such as ImageNet, to learn features that are useful for various computer vision tasks.
`self.model = ...`: The model instance is assigned to the `self.model` attribute. This suggests that this line of code is part of a class definition in Python, and `self` refers to an instance of that class.

1. `self.model.classifier[-1]`: This part accesses the last layer of the classifier. The classifier of VGG-19 is typically a series of fully connected (linear) layers. The `[-1]` indexing is used to get the last layer in the classifier.
2. `.in_features`: This attribute returns the number of input features for a linear layer. In the context of a neural network, the input features represent the number of units in the preceding layer. For the last layer in the classifier, `in_features` gives you the number of input features that the layer expects.

So, `in_features` will be the number of input features expected by the last fully connected layer of the VGG-19 model. This information is often used when defining or modifying subsequent layers in a neural network architecture.

The `[:-1]` syntax is a slice notation in Python used to select all elements of a list or sequence except the last one

Explanation of Model.children:
`self.model.classifier.children()`

```
child_counter = 0
for child in model.children():
  print(" child", child_counter, "is:")
  print(child)
  child_counter += 1
```

The above function gives us the following:

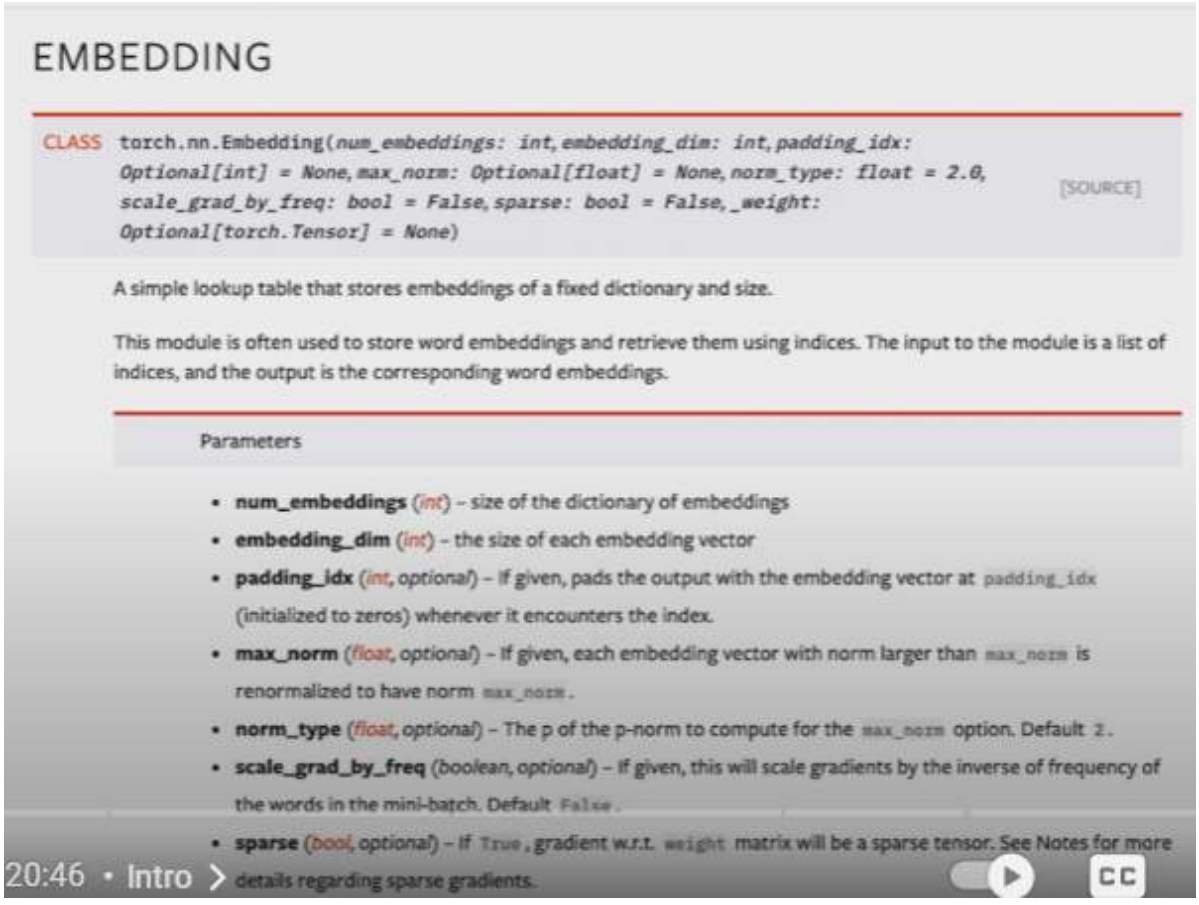
child 0 is -
Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
child 1 is:
BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True)
child 2 is:

ReLU (inplace)
child 3 is:
MaxPool2d (size=(3, 3), stride=(2, 2), padding=(1, 1), dilation=(1, 1))
child 4 is:
Sequential (
(0): BasicBlock ((conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,1), bias=False)
(bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True)
(relu): ReLU (inplace)
(conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,1), bias=False)
(bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True)
)
(1): BasicBlock ((conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,1), bias=False)
(bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True)
(relu): ReLU (inplace)
(conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,1), bias=False)
(bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True)
)
)
child 5 is:
Sequential (
(0): BasicBlock ((conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1,1),bias=False)
(bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True)
(relu): ReLU (inplace)
(conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
(bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True)
(downsample): Sequential (
(0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
(1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True)
)
)
(1): BasicBlock (
(conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
(bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True)

(relu): ReLU (inplace)
(conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
(bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True)
)
)
child 6 is:
others are not shown because of consuming less space

Basically it tells us all about the layers present in the pretrained model.

What is nn.Embedding?

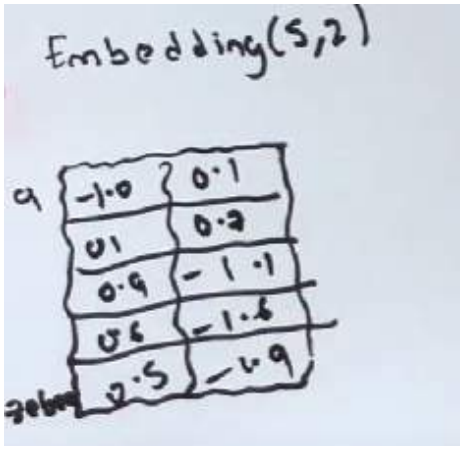


This is how it looks :

```
In [2]: from torch.nn import Embedding

In [3]: n_embeddings, dim = 10, 4

In [4]: emb_1 = Embedding(n_embeddings, dim)

In [5]: emb_1
Out[5]: Embedding(10, 4)

In [6]: emb_1.weight
Out[6]:
Parameter containing:
tensor([[-1.1031,  1.2085, -1.4763, -3.0549],
        [-1.2877,  0.3131, -1.3447,  0.5349],
        [-0.9851, -0.7829,  2.1659,  0.8222],
        [-1.3978, -1.0284, -3.0019, -0.1105],
        [ 0.7094, -0.8497, -1.4942,  0.3085],
        [-0.5379,  0.9219, -1.1207,  0.4040],
        [-0.1737, -0.7919, -0.6578, -1.2517],
        [ 1.0310,  1.0146,  0.4159, -0.4621],
        [-0.9632,  0.0950, -1.8883,  0.8733],
        [ 1.6225,  0.2400,  1.5813, -2.0829]], requires_grad=True)
```

requires_grad=true means that the entire array is learnable

We can represent a word embedding as shown below :



Word Embeddings:

It is an approach for representing words and documents. Word Embedding or Word Vector is a numeric vector input that represents a word in a lower-dimensional space. It allows words with similar meanings to have a similar representation. They can also approximate meaning. A word vector with 50 values can represent 50 unique features

Features: Anything that relates words to one another. E.g.: Age, Sports, Fitness, Employed, etc. Each word vector has values corresponding to these features.

How are Word Embeddings used?

- They are used as input to machine learning models.
  Take the words —-> Give their numeric representation —-> Use in training or inference
- To represent or visualize any underlying patterns of usage in the corpus that was used to train them.

Implementations of Word Embeddings:

Word Embeddings are a method of extracting features out of text so that we can input those features into a machine learning model to work with text data. They try to preserve syntactical and semantic information.We can get a sparse matrix if most of the elements are zero. Large input vectors will mean a huge number of weights which will result in high computation required for training

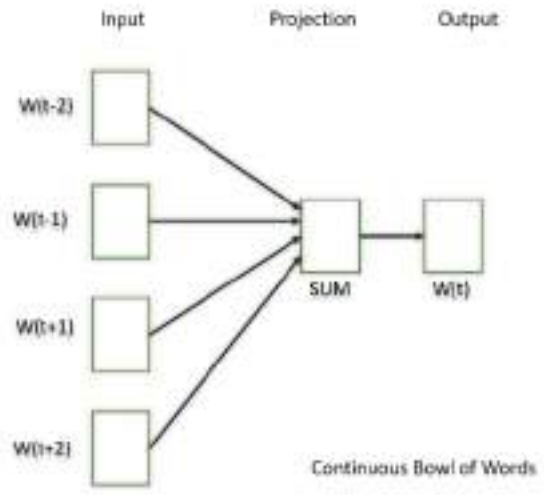
Understanding Word2Vec:

In Word2Vec every word is assigned a vector. We start with either a random vector or **one-hot vector**.

One-Hot vector: A representation where only one bit in a vector is 1.If there are 500 words in the corpus then the vector length will be 500. After assigning vectors to each word we take a window size and iterate through the entire corpus. While we do this there are two **neural embedding methods** which are used:

1.1) Continuous Bag of Words(CBOW)

In this model what we do is we try to fit the neighboring words in the window to the central word.

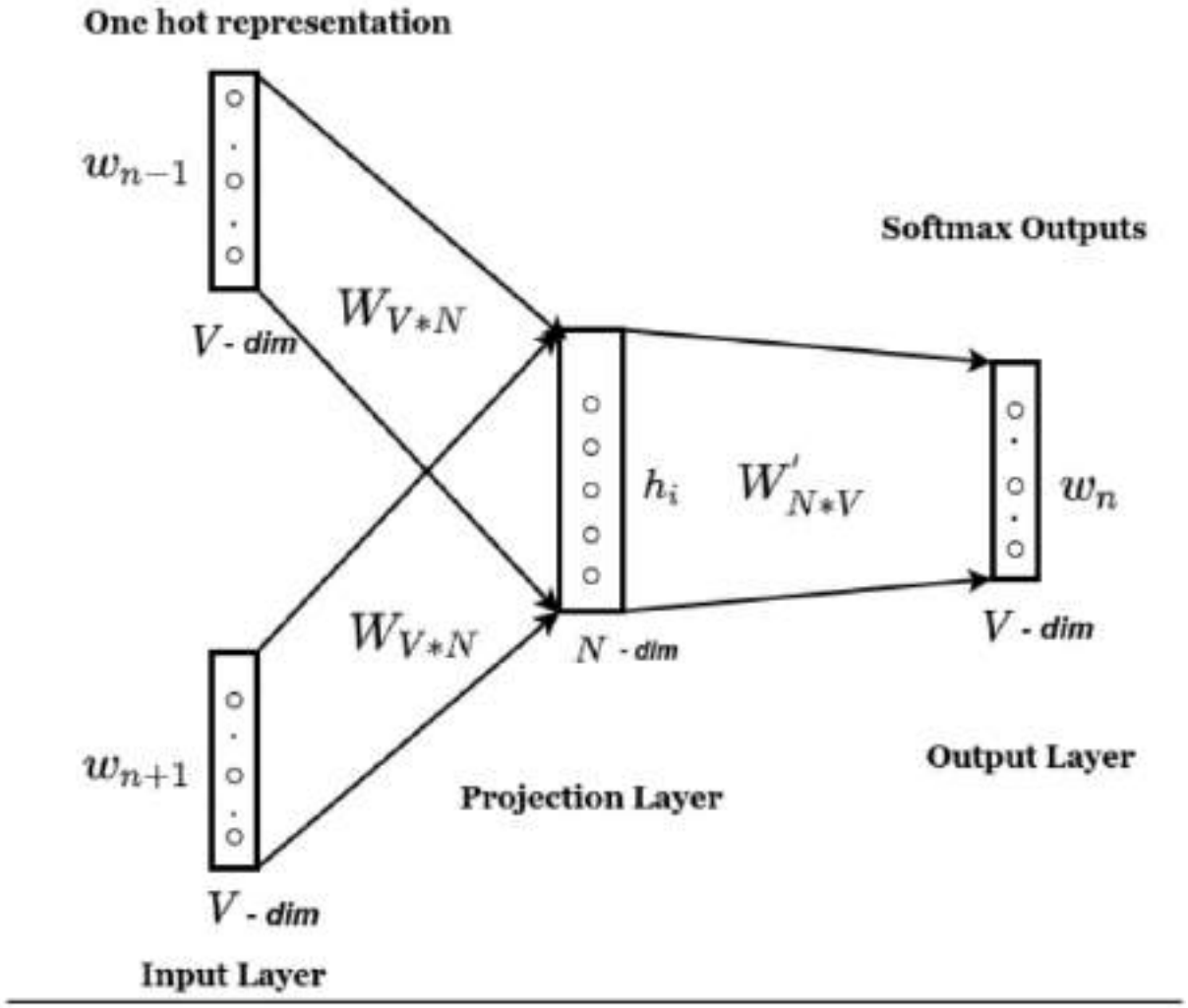


The Continuous Bag of Words (CBOW) model is a popular method for training word embeddings, which are representations of words in a numerical vector space,The goal of CBOW is to predict a target word given the context words in a sentence,The input layer is used to represent the context words, the hidden layer is used to learn the word embeddings, and the output layer is used to predict the target word,The input layer is typically represented by a one-hot encoded vector, where each element in the vector corresponds to a specific word in the vocabulary. For example, if the vocabulary contains 10,000 words, the input layer will have 10,000 elements

The hidden layer is where the word embeddings are learned. It is a dense layer, with each neuron representing a specific word in the vocabulary. The number of neurons in the hidden layer is the same as the number of words in the vocabulary.

The output layer is also a dense layer, with each neuron representing a specific word in the vocabulary. The number of neurons in the output layer is the same as the number of words in the vocabulary:

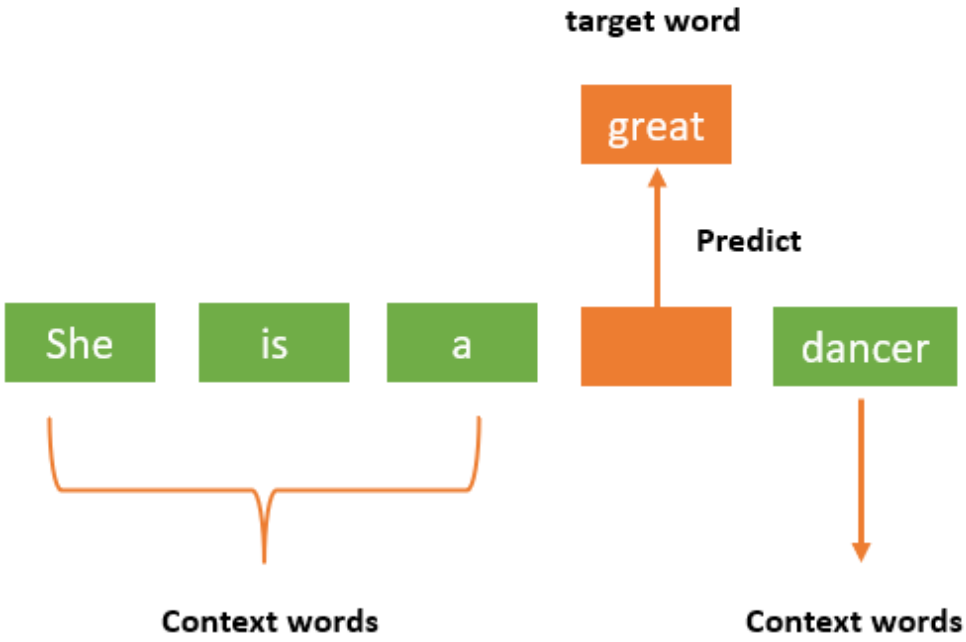
- Let x(1), x(2), ..., x(n) be the context words, where x(i) is a one-hot encoded vector.
- Let W(1) be the weight matrix connecting the input layer to the hidden layer, and W(2) be the weight matrix connecting the hidden layer to the output layer.
- Let h be the hidden layer, which is the average of the input vectors, h = 1/n * (x(1) + x(2) + ... + x(n))
- Let y be the output layer, which is the probability distribution over the vocabulary, y = softmax(W(2) * h)
- The target word is selected as the word with the highest probability in y.



Continuous Bag of Words (CBOW) model

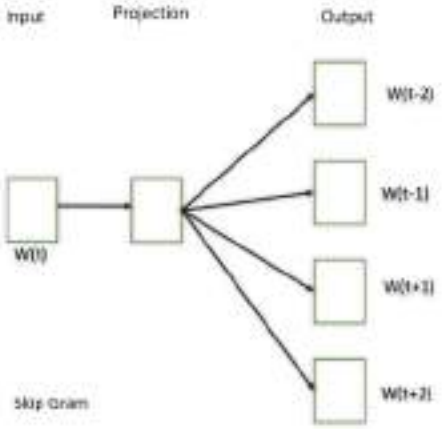
The training process for CBOW is to minimize the difference between the predicted probability distribution y and the actual target word using a loss function such as cross-entropy loss. This process is repeated for each sentence in the training dataset, resulting in the learned embeddings.One of the key advantages of CBOW is its efficiency. Because the model only needs to predict a single target word given a set of context words, it can train on a much larger dataset than the skip-gram model, which needs to predict multiple context words given a target word

Word2vec is a neural network-based method for generating word embeddings, which are dense vector representations of words that capture their semantic meaning and relationships


target word
great
Predict
She
is
a
dancer
Context words
Context words

1.2) Skip Gram

In this model, we try to make the central word closer to the neighboring words. It is the complete opposite of the CBOW model. It is shown that this method produces more meaningful embeddings.



After applying the above neural embedding methods we get trained vectors of each word after many iterations through the corpus. These trained vectors preserve syntactical or semantic information and are converted to lower dimensions. The vectors with similar meaning or semantic information are placed close to each other in space.

Architecture of the CBOW model

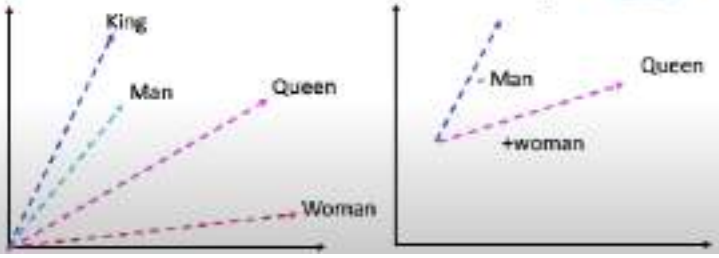
The CBOW model uses the target word around the context word in order to predict it. Consider the above example **"She is a great dancer."** The CBOW model converts this phrase into pairs of context words and target words. The word pairings would appear like this **([she, a], is), ([is, great], a) ([a, dancer], great)** having window size=2.

General Overview

- Map word into vector space
- Makes the words that have a similar context to have similar embeddings
- Thus, in this vector space, these words are close.

Each word is transformed into a distributed representation of its weight across all Other words

	King	Man	Queen	Woman
Royal	0.99	0.2	0.99	0.3
Masculinity	0.99	0.99	0.05	0.05
Femineity	0.05	0.05	0.99	0.99
Age	0.7	0.7	0.6	0.6



How It Works?

Context Word

- Learn the context in which a particular word appears
- Words that occurs in similar context have similar embedding

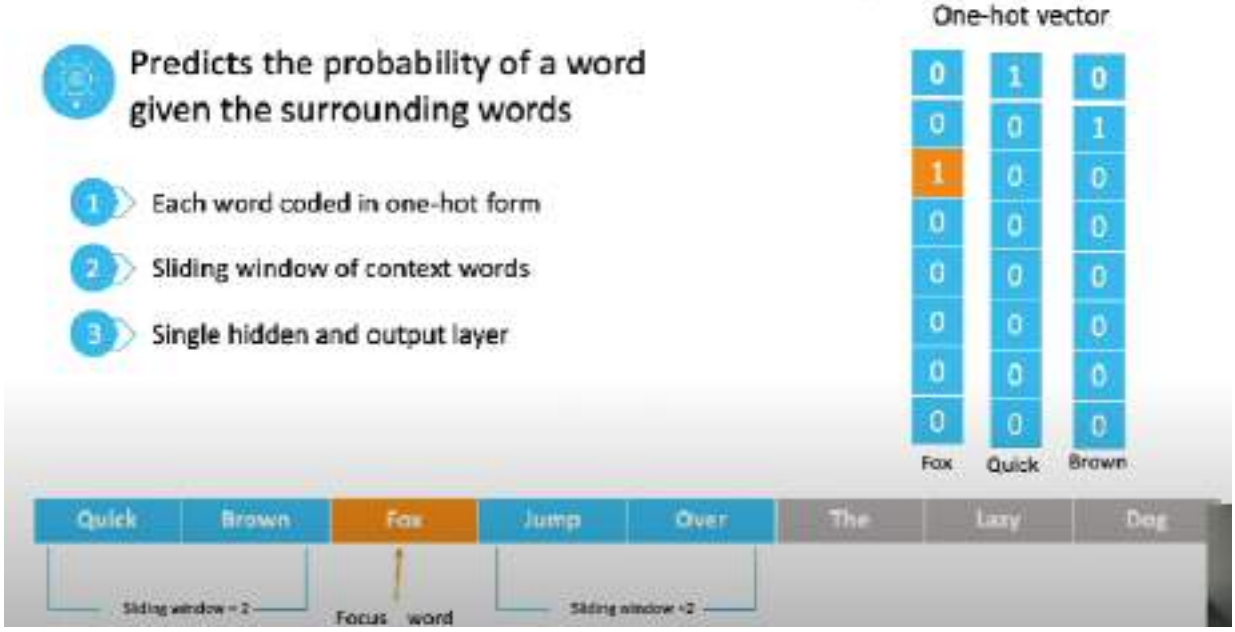
Uses two algorithms:

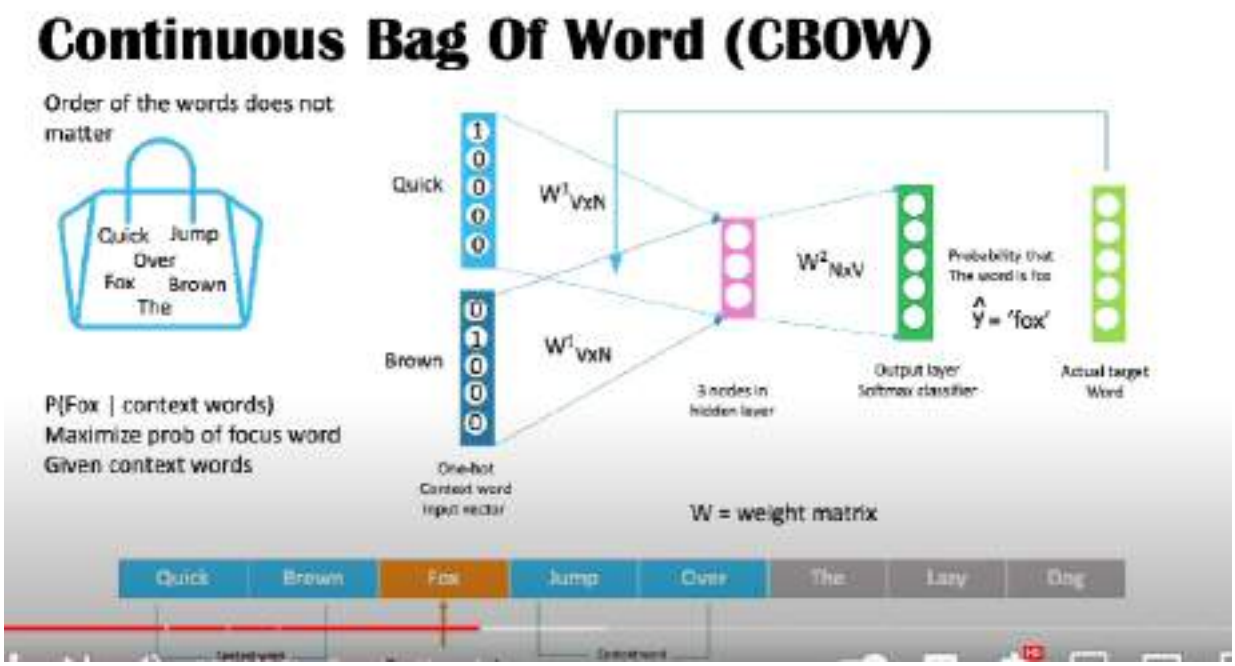
1. Focus word » Continuous Bag Of Word (CBOW)
2. Context word » Skip- Gram

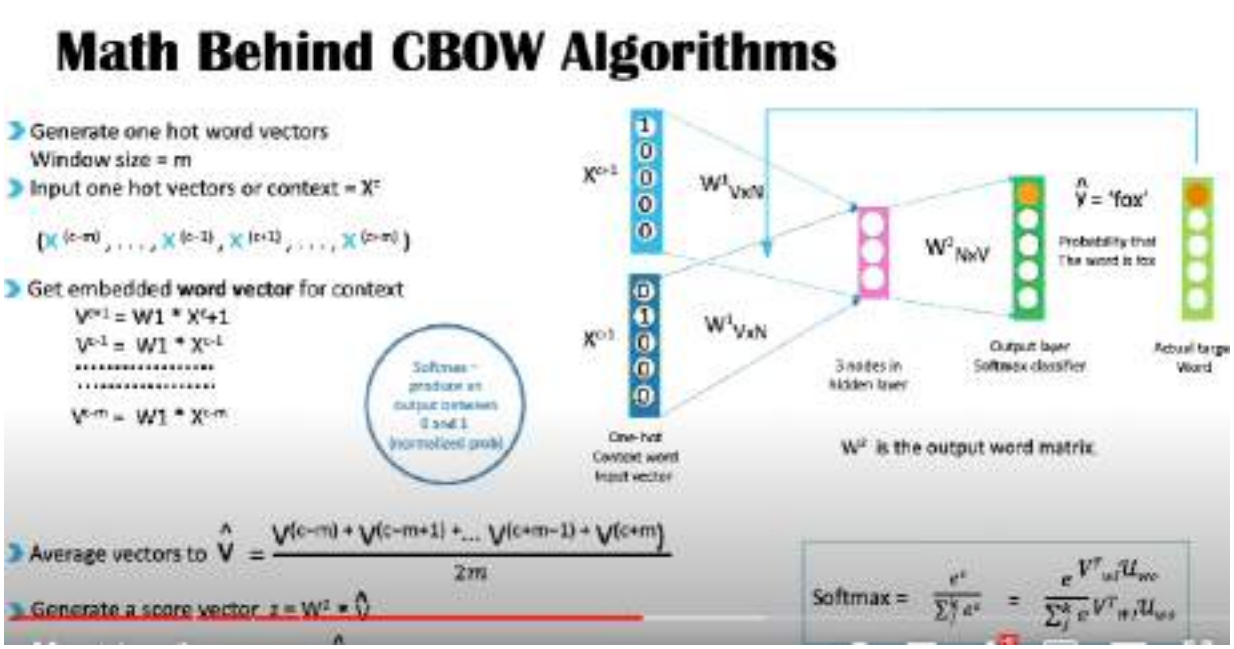
Continuous Bag Of Word (CBOW)

Predicts the probability of a word given the surrounding words

1. Each word coded in one-hot form
2. Sliding window of context words
3. Single hidden and output layer



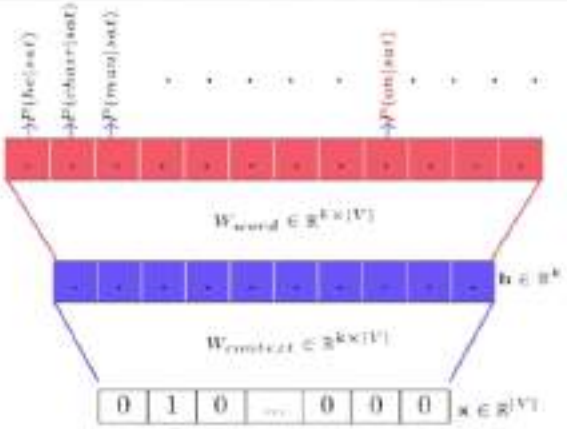




NPTEL notes on CBOW:

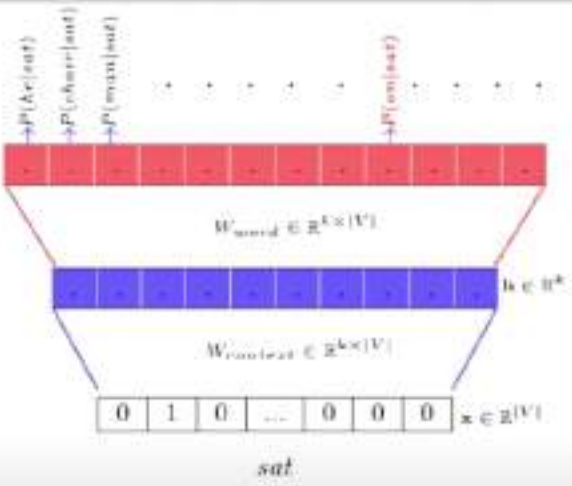
Sometime in the 21st century, Joseph Cooper, a widowed farmer, engineer and former NASA pilot, runs a farm with his father-in-law Donald, son Tom, and daughter Murphy. It is post-truth society (Cooper is reprimanded for telling Murphy that the Apollo missions did indeed happen) and a series of crop blights threatens humanity's survival. Murphy believes her bedroom is haunted by a poltergeist. When a pattern is created out of dust on the floor, Cooper realizes that gravity is behind its formation, not a "ghost". He interprets the pattern as a set of geographic coordinates formed into binary code. Cooper and Murphy follow the coordinates to a secret NASA facility, where they are met by Cooper's former professor, Dr. Brand.

- **Consider this Task:** Predict n-th word given previous n-1 words
- **Example:** he sat on a chair
- **Training data:** All n-word windows in your corpus
- Training data for this task is easily available (take all n word windows from the whole of wikipedia)
- For ease of illustration, we will first focus on the case when $n = 2$ (*i.e.*, predict second word based on first word)



sat

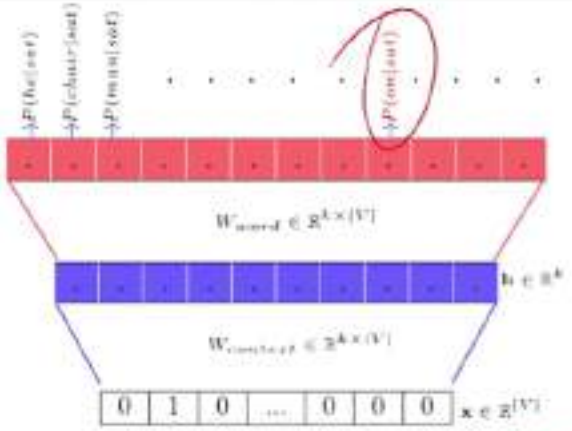
- We will model this problem using a feedforward neural network
- **Input:** One-hot representation of the **context word**
- **Output:** There are $|V|$ words (classes) possible and we want to predict a probability distribution over these $|V|$ classes (multi-class classification problem)
- **Parameters:** $\mathbf{W}_{context} \in \mathbb{R}^{k \times |V|}$ and $\mathbf{W}_{word} \in \mathbb{R}^{k \times |V|}$
  (we are assuming that the set of **words** and **context** words is the same: each of size $|V|$)



- What is the product $\mathbf{W}_{context}\mathbf{x}$ given that $\mathbf{x}$ is a one hot vector
- It is simply the i-th column of $\mathbf{W}_{context}$

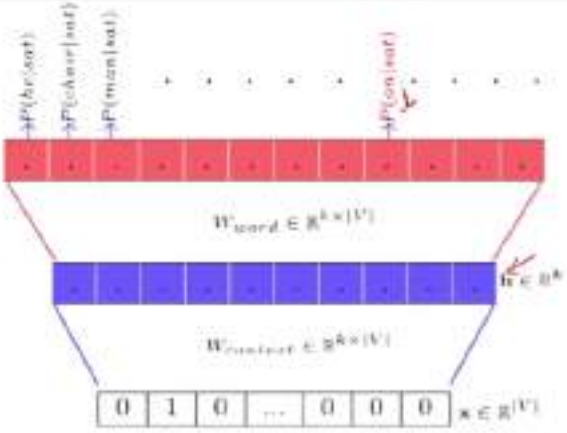
$$\begin{bmatrix} -1 & 0.5 & 2 \\ 3 & -1 & -2 \\ -2 & 1.7 & 3 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.5 \\ -1 \\ 1.7 \end{bmatrix}$$

- So when the i^{th} word is present the i^{th} element in the one hot vector is ON and the i^{th} column of $\mathbf{W}_{context}$ gets selected
- In other words, there is a one-to-one correspondence between the words and the columns of $\mathbf{W}_{context}$
- More specifically, we can treat the i-th column of $\mathbf{W}_{context}$ as the rep context i



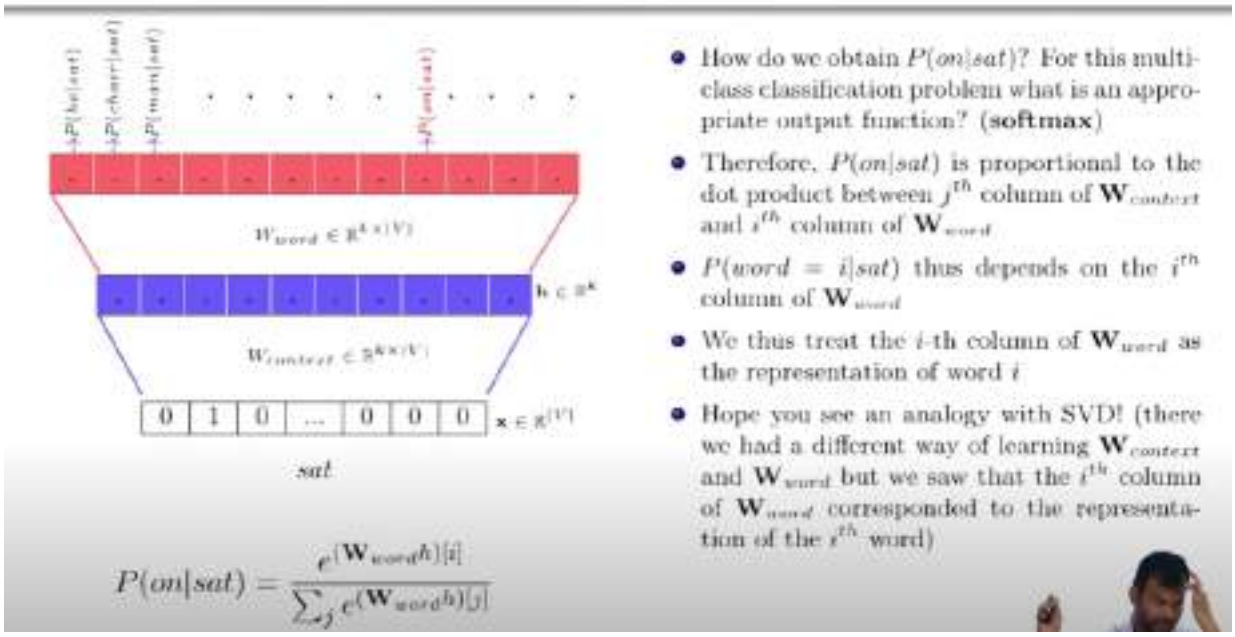
- How do we obtain $P(on|sat)$? For this multi-class classification problem what is an appropriate output function? (**softmax**)

$$P(on|sat) = \frac{e^{(\mathbf{W}_{word}h)[i]}}{\sum_j e^{(\mathbf{W}_{word}h)[j]}}$$

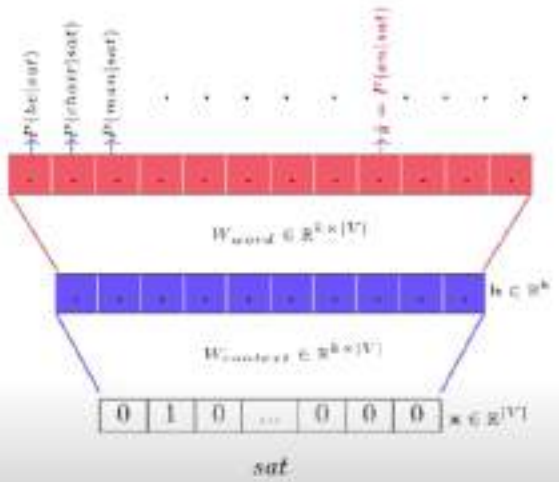


- How do we obtain $P(on|sat)$? For this multi-class classification problem what is an appropriate output function? (**softmax**)
- Therefore, $P(on|sat)$ is proportional to the dot product between j^{th} column of $\mathbf{W}_{context}$ and i^{th} column of $\mathbf{W}_{word}$

$$P(on|sat) = \frac{e^{(\mathbf{W}_{word}h)[i]}}{\sum_j e^{(\mathbf{W}_{word}h)[j]}}$$



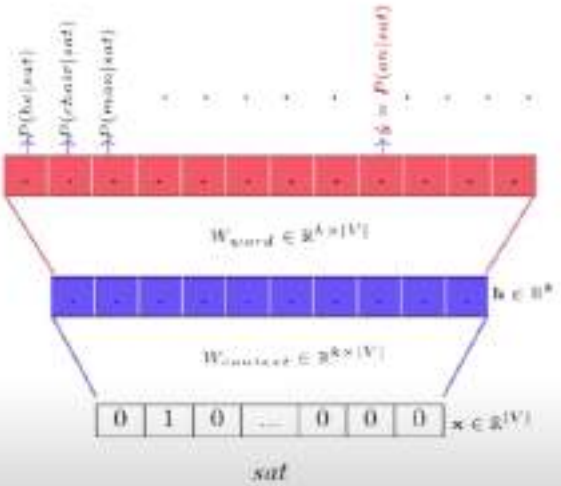
The training algorithm in the above is backprogation and the loss function is cross entropy.



- We denote the context word (sat) by the index c and the correct output word (on) by the index w
- For this multiclass classification problem what is an appropriate output function ($\hat{y} = f(x)$) ? **softmax**
- What is an appropriate loss function? **cross entropy**

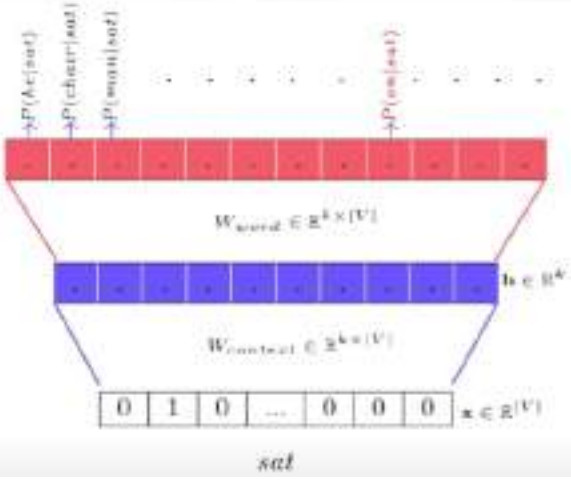
$$\mathscr{L}(\theta) = -\log \hat{y}_w = -\log P(w|c)$$
$$h = W_{context} \cdot x_c = u_c$$
$$\hat{y}_w = \frac{exp(u_c \cdot v_w)}{\sum_{w' \in V} exp(u_c \cdot v_{w'})}$$

u_c is the column of $W_{context}$ corresponding to context word c and v_w is the column of W_{word} corresponding to the word w



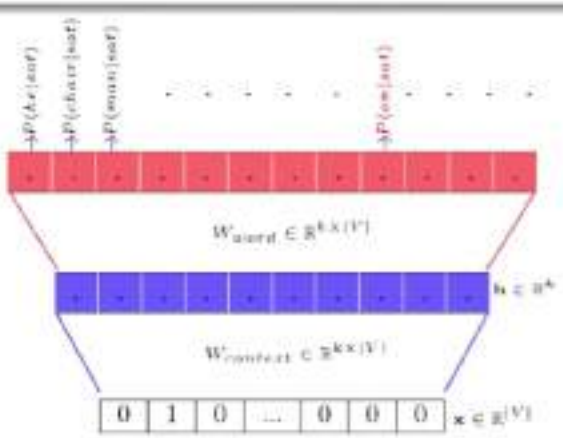
- We denote the context word (sat) by the index c and the correct output word (on) by the index w
- For this multiclass classification problem what is an appropriate output function ($\hat{y} = f(x)$) ? **softmax**
- What is an appropriate loss function? **cross entropy**

$$\mathscr{L}(\theta) = -\log \hat{y}_w = -\log P(w|c)$$
$$h = W_{context} \cdot x_c = u_c$$
$$\hat{y}_w = \frac{exp(u_c \cdot v_w)}{\sum_{w' \in V} exp(u_c \cdot v_{w'})}$$



- How do we train this simple feed forward neural network? backpropagation
- Let us consider one input-output pair (c, w) and see the update rule for v_w

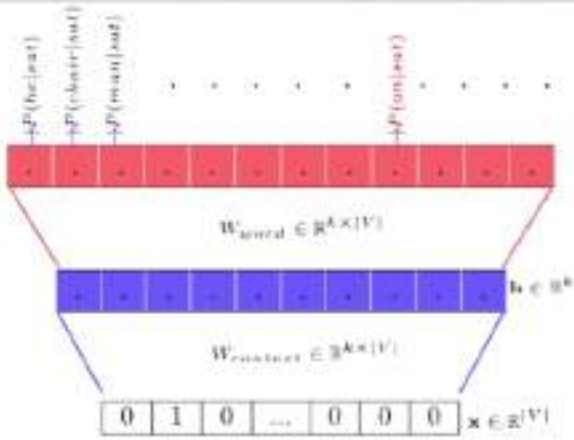
We are learning the matrices Wword and Wcontext as shown below :



sat

$$\nabla_{v_w} = \frac{\partial}{\partial v_w} \mathscr{L}(\theta)$$

$$\begin{aligned}
\mathscr{L}(\theta) &= -\log \hat{y}_w \\
&= -\log \frac{exp(u_c \cdot v_w)}{\sum_{w' \in V} exp(u_c \cdot v_{w'})} \\
&= -(u_c \cdot v_w - \log \sum_{w' \in V} exp(u_c \cdot v_{w'})) \\
\nabla_{v_w} &= -(u_c - \frac{exp(u_c \cdot v_w)}{\sum_{w' \in V} exp(u_c \cdot v_{w'})} \cdot u_c)
\end{aligned}$$

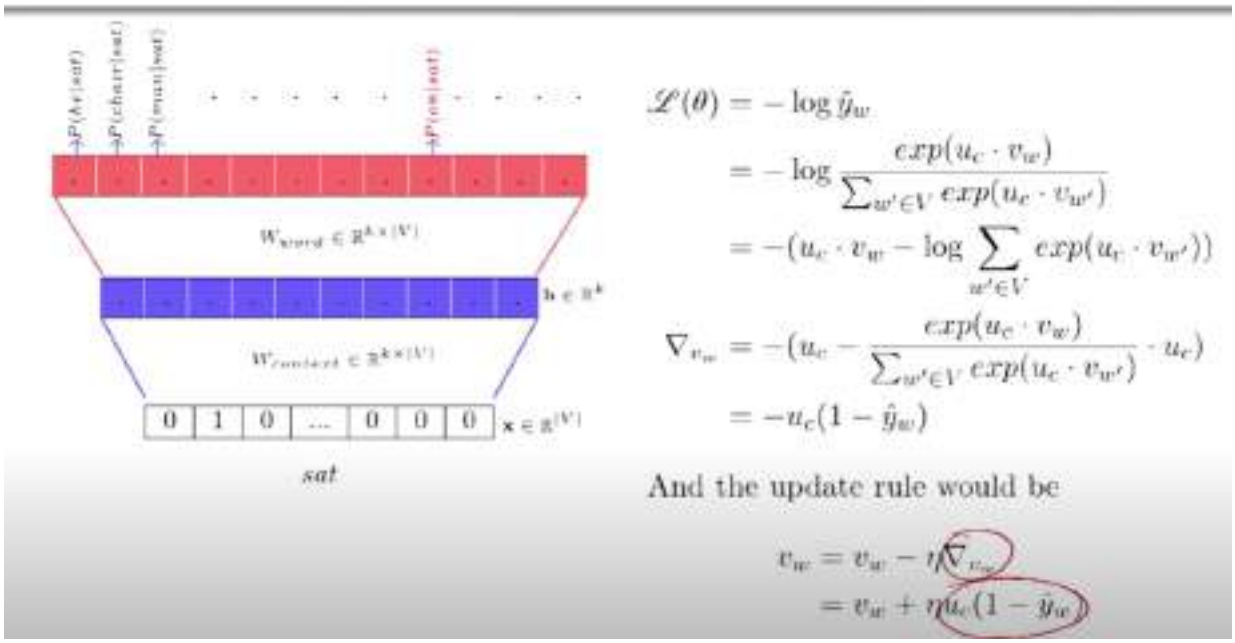


sat

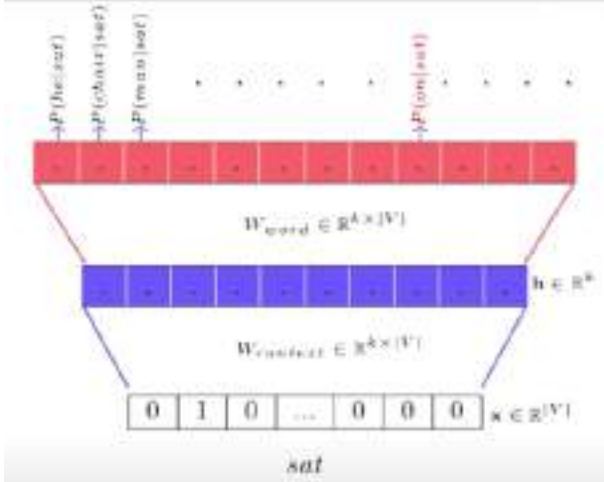
$$\begin{aligned}
\mathscr{L}(\theta) &= -\log \hat{y}_w \\
&= -\log \frac{exp(u_c \cdot v_w)}{\sum_{w' \in V} exp(u_c \cdot v_{w'})} \\
&= -(u_c \cdot v_w - \log \sum_{w' \in V} exp(u_c \cdot v_{w'})) \\
\nabla_{v_w} &= -(u_c - \frac{exp(u_c \cdot v_w)}{\sum_{w' \in V} exp(u_c \cdot v_{w'})} \cdot u_c) \\
&= -u_c(1 - \hat{y}_w)
\end{aligned}$$

And the update rule would be

$$v_w = v_w - \eta \nabla_{v_w}$$



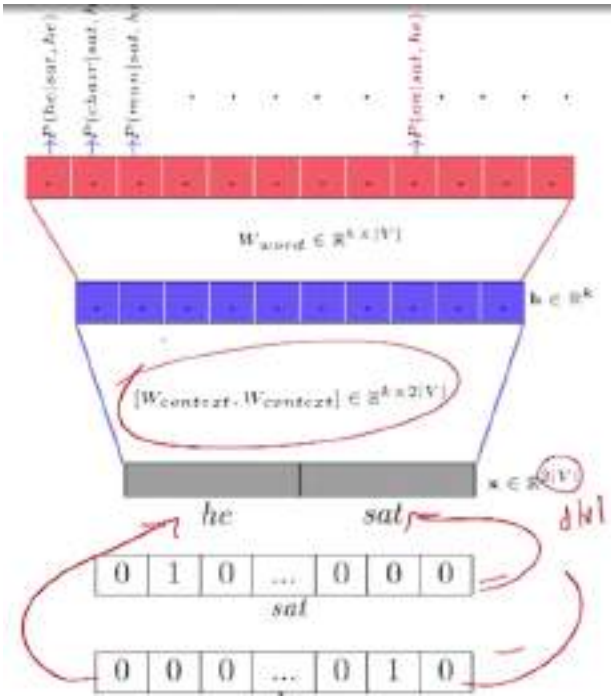
In the update rule above we are trying to make the word representation closer to the context representation.



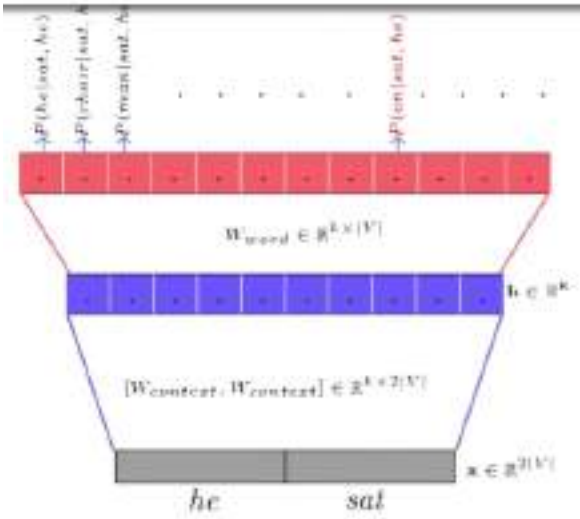
- This update rule has a nice interpretation

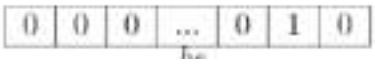
$$v_w = v_w + \eta u_c (1 - \hat{y}_w)$$

- If $\hat{y}_w \rightarrow 1$ then we are already predicting the right word and v_w will not be updated
- If $\hat{y}_w \rightarrow 0$ then v_w gets updated by adding a fraction of u_c to it
- This increases the cosine similarity between v_w and u_c (How? Refer to slide 38 of Lecture 2)
- The training objective ensures that the cosine similarity between word (v_w) and context word (u_c) is maximized



- In practice, instead of window size of 1 it is common to use a window size of d





- In practice, instead of window size of 1 it is common to use a window size of d
- So now,

$$h = \sum_{i=1}^{d-1} u_{c_i}$$

- $[W_{context}, W_{context}]$ just means that we are stacking 2 copies of $W_{context}$ matrix

$$\begin{bmatrix} -1 & 0.5 & 2 & -1 & 0.5 & 2 \\ 3 & -1.0 & -2 & 3 & -1.0 & -2 \\ -2 & 1.7 & 3 & -2 & 1.7 & 3 \end{bmatrix} \begin{matrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{matrix} \begin{matrix} \\ \} \text{ sat} \\ \\ \\ \\ \} \text{ he} \end{matrix}$$

- In practice, instead of window size of 1 it is common to use a window size of d
- So now,

$$h = \sum_{i=1}^{d-1} u_{c_i}$$

- $[W_{context}, W_{context}]$ just means that we are stacking 2 copies of $W_{context}$ matrix

$$\begin{bmatrix} -1 & 0.5 & 2 & -1 & 0.5 & 2 \\ 3 & -1.0 & -2 & 3 & -1.0 & -2 \\ -2 & 1.7 & 3 & -2 & 1.7 & 3 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \begin{matrix} \\ \} sat \\ \\ \\ \\ \} he \end{matrix}$$

$$= \begin{bmatrix} 2.5 \\ -3.0 \\ 4.7 \end{bmatrix}$$

Question Encoder:
1.nn.Embedding:You create an `nn.Embedding` layer by specifying the size of the vocabulary (number of unique items) and the dimensionality of the embedding space. For example, if you're working with a vocabulary of 10,000 words and want each word to be represented as a 100-dimensional vector, you'd initialize `nn.Embedding(10000, 100)`

Let us take the example of :
Let's say we're working with a simple vocabulary of three words: "cat," "dog," and "bird," and we want to represent them as 2-dimensional vectors.

```
import torch
import torch.nn as nn

# Define the vocabulary size and embedding dimension
vocab_size = 3
embedding_dim = 2

# Create an nn.Embedding layer
embedding_layer = nn.Embedding(vocab_size, embedding_dim)

# Define a sequence of word indices
word_indices = torch.LongTensor([0, 1, 2])  # Indices for "cat," "dog," and "bird"

# Pass the word indices through the embedding layer
word_embeddings = embedding_layer(word_indices)

# Print the results
print(word_embeddings)
```

In this example:

- `vocab_size` is 3 because we have three words in our vocabulary.
- `embedding_dim` is 2, meaning each word will be represented as a 2-dimensional vector.
- `word_indices` is a tensor containing the indices of the words we want to embed (0 for "cat," 1 for "dog," and 2 for "bird").

When we pass `word_indices` through the `embedding_layer`, it looks up the embeddings for these indices and returns the corresponding vectors. The output might look something like:

lua — Copy code

```
tensor([[-0.1234,  0.6678],
        [ 1.2345, -2.3456],
        [-0.9876,  2.8765]], grad_fn=<EmbeddingBackward>)
```

Notes from the Paper:

- In this paper they have combined LSTM with CNN to generate a text
- In their model, the LSTM question representation is conditioned on the CNN image features at each time step, and the final LSTM hidden state is used to sequentially decode the answer phrase
- the model developed in this paper explores “late fusion” – i.e., the LSTM question representation and the CNN image features are computed independently, fused via an element-wise multiplication, and then passed through fullyconnected layers to generate a softmax distribution over output answer classes
-

The Vgg19 has the following architecture:

VGG19

So in simple language VGG is a deep CNN used to classify images. The layers in VGG19 model are as follows:

- Conv3x3 (64)
- Conv3x3 (64)
- MaxPool
- Conv3x3 (128)
- Conv3x3 (128)
- MaxPool
- Conv3x3 (256)
- Conv3x3 (256)
- Conv3x3 (256)
- Conv3x3 (256)
- MaxPool
- Conv3x3 (512)
- Conv3x3 (512)
- Conv3x3 (512)
- Conv3x3 (512)
- MaxPool
- Conv3x3 (512)

- Conv3x3 (512)
- Conv3x3 (512)
- Conv3x3 (512)
- Conv3x3 (512)
- MaxPool
- Conv3x3 (512)
- Conv3x3 (512)
- Conv3x3 (512)
- Conv3x3 (512)
- MaxPool
- Fully Connected (4096)
- Fully Connected (4096)
- Fully Connected (1000)
- SoftMax

---

json file:A JSON file stores data in key-value pairs and arrays; the software it was made for then accesses the data. JSON allows developers to store various data types as human-readable code, with the keys serving as names and the values containing related data

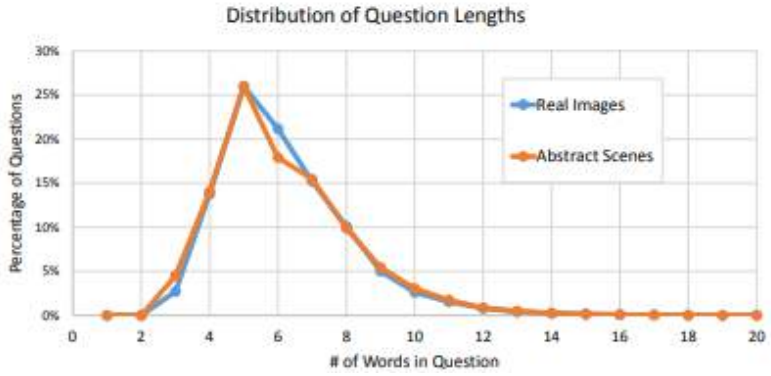
JSON syntax is derived from JavaScript object notation syntax:

- Data is in key/value pairs
- Data is separated by commas
- Curly braces hold objects
- Square brackets hold arrays

VQA Dataset Analysis:
Types of Question:

- Given the structure of questions generated in the English language, we can cluster questions into different types based on the words that start the question.
- The distribution of questions based on the first four words of the questions for both the real images (left) and abstract scenes (right).

- There exists a surprising variety of question types, including “What is. . .”, “Is there. . .”, “How many. . .”, and “Does the. . .”.
- A particularly interesting type of question is the “What is. . .” questions, since they have a diverse set of possible answer.
- Lengths of questions:We see that most questions range from four to ten words



Answers:

- . We can see that a number of question types, such as “Is the. . . ”, “Are. . . ”, and “Does. . . ” are typically answered using “yes” and “no” as answer
- Other questions such as “What is. . . ” and “What type. . . ” have a rich diversity of response.
- Other question types such as “What color. . . ” or “Which. . . ” have more specialized responses, such as colors, or “left” and “right”
- Lengths of the answers:Most answers consist of a single word, with the distribution of answers containing one, two, or three words, respectively being 89.32%, 6.91%, and 2.74% for real images and 90.51%, 5.89%, and 2.49% for abstract scene

What is important in this analysis?
The brevity of answers is not surprising, since the questions tend to elicit specific information from the image,This is in contrast with image captions that generically describe the entire image and hence tend to be longer. The brevity of our answers makes automatic evaluation feasible

Many questions are answered using either “yes” or “no” (or sometimes “maybe”) – 38.37% and 40.66% of the questions on real images and abstract scenes respectively. Among these ‘yes/no’ questions, there is a bias towards “yes” – 58.83% and 55.86% of ‘yes/no’ answers are “yes” for real images and abstract scenes

Question types such as “How many. . . ” are answered using numbers – 12.31% and 14.48% of the questions on real images and abstract scenes are ‘number’ questions. “2” is the most popular answer among the ‘number’ questions, making up 26.04% of the ‘number’ answers for real images and 39.85% for abstract scenes

Subject Confidence. When the subjects answered the questions, we asked “Do you think you were able to answer the question correctly?”. Fig. 6 shows the distribution of responses. A majority of the answers were labeled as confident for both real images and abstract scenes

In order to sort out different answers or opinions we take 10 answers for each question

answers that are “a brief phrase and not a complete sentence"

avoid using conversational language or inserting your opinion

The candidates were also asked if they were able Do you think you were able to answer the question correctly and were given the choice of given the choices of “no”, “maybe”, and “yes”.

In order to see if the answers are correct than we do the following:

For the open-ended task, the generated answers are evaluated using the following accuracy metric:

$$\text{accuracy} = \min\left(\frac{\#\text{ humans that provided that answer}}{3}, 1\right)$$

an answer is deemed 100% accurate if at least 3 workers provided that exact answer.2 Before comparison, all responses are made lowercase, numbers converted to digits, and punctuation & articles removed.

We avoid using soft metrics such as Word2Vec [39], since they often group together words that we wish to distinguish, such as “left” and “right”. We also avoid using evaluation metrics from machine translation such as BLEU and ROUGE because such metrics are typically applicable and reliable for sentences containing multiple words.

BLOCK Method:

Linear Modals:, linear models constitute a handful building block to transform the input x and to match it with the desired output y.

Bilinear Models:When dealing with two modalities, not only do we need to properly transform each input x1 and x2 into a representation that fits the problem, we also want to model the interactions between these modalities.. A natural candidate to extend linear models for two inputs are bilinear models.

A bilinear method is defined with the help of a tensor When the input dimensions grow (thousand or more), learning a full tensor T becomes quickly intractable. The main issue is to reduce and control the numbers of parameters representing T.

Block :Block Superdiagonal Fusion framework for multimodal representation based on the block-term tensor decomposition

BLOCK enables to model very rich (i.e. full bilinear) interactions between groups of features, while the block structure limits the whole complexity of the model, which enables to keep expressive (i.e. high dimensional) mono-modal representations.

How is it working in Block method?
Model detailed interactions between different pieces of information (blocks) in a way that captures fine details. Imagine you have a complex puzzle, and each piece represents a different aspect of the task. They want BLOCK to be good at understanding how these pieces interact with each other in a very detailed way.
BLOCK to become too complicated with lots of parameters, which are like the settings or characteristics it learns from the data. Too many parameters can make things slow and less practical.
the plan is to adjust the number of these puzzle pieces (blocks) and the size of each piece in BLOCK to find that sweet spot. It's like tweaking the puzzle to have enough pieces to capture all the important details but not too many that it becomes overwhelming.
What do you mean by projection?
A "projection" refers to a way of transforming or mapping data from one space to another. It's like shining a light on an object and capturing its shadow—a simplified representation that retains essential information.
In the BLOCK framework, these projections are applied to the blocks of information. Each block represents a different aspect of the data, and the projection is a method that transforms this block into a form that helps capture important patterns or relationships. Think of it as taking a 3D object and projecting its shadow onto a 2D surface. The shadow might not capture all the intricate details, but it gives you a useful representation that retains the overall structure.
So, when the paper talks about adapting the size of each projection, it means adjusting how much information is retained in this simplified representation. It's a way of fine-tuning how much detail you want to keep from each block of information while managing the overall complexity of the model
The maths behind:
A bilinear model is a way of combining two sets of numbers (vectors), let's call them x1 and x2. The goal is to mix them up in a specific way to get a new set of numbers, y.

This mixing is done using something called tensor products. It's a bit like a mathematical recipe that takes x1 and x2, stirs them together, and produces y.
Each number in the resulting set y is calculated using a fancy formula (quadratic form). For every number (yk) in y, you're doing some math involving the elements of x1 and x2, and the magic ingredient is a tensor T
This tensor T is like a big table of numbers with three dimensions (I, J, and K). The values in this table determine how much each element in x1 and x2 contributes to the final result y
So, in short, a bilinear model is like a cooking recipe where you mix two sets of ingredients (x1 and x2) using a special table (tensor T), and you get a new dish (y) with some math magic
Imagine you have a big mathematical object T that determines how to mix two sets of numbers (x1 and x2) to get another set of numbers (y). Instead of dealing with this big T directly, the paper suggests breaking it down into smaller, more manageable parts.
So, T is expressed as a sum of simpler blocks, each represented by a set of matrices (D, A, B, and C). These matrices are like building blocks that, when combined, recreate the original T.
Now, these building blocks are structured in a way that allows for a neat simplification. The paper uses a block-term decomposition, which basically means breaking down T into smaller blocks arranged in a structured manner.
This structured arrangement makes the math more manageable, and it's like having a recipe that's easier to follow. The resulting y, which is the set of numbers you're interested in, is then expressed using these simplified blocks.
Decomposition of the Big Learning Tensor into smaller simpler parts:

Imagine you have a big mathematical object T that determines how to mix two sets of numbers (x1 and x2) to get another set of numbers (y). Instead of dealing with this big T directly, the paper suggests breaking it down into smaller, more manageable parts.
So, T is expressed as a sum of simpler blocks, each represented by a set of matrices (D, A, B, and C). These matrices are like building blocks that, when combined, recreate the original T
Now, these building blocks are structured in a way that allows for a neat simplification. The paper uses a block-term decomposition, which basically means breaking down T into smaller blocks arranged in a structured manner.
This structured arrangement makes the math more manageable, and it's like having a recipe that's easier to follow. The resulting y, which is the set of numbers you're

interested in, is then expressed using these simplified blocks.
This is how we break down the learning Tensor T :

1. Tensor T Decomposition:

$$T = \sum_{r=1}^{R} D_r \times_1 A_r \times_2 B_r \times_3 C_r$$

This equation says that T is a sum of R terms, each involving matrices D, A, B, and C. The index r indicates each term in the sum.
The use of the index in the block-term decomposition serves a crucial purpose—it allows for a decomposition of the tensor into a sum of simpler components. This decomposition helps reduce the complexity of the model and provides a more structured and manageable representation.

Summation of Components:

- The term $\sum_{r-1}^{R}$ indicates that you are summing up individual components from $r = 1$ to R.
- Each r-th component, represented by $D_r \times_1 A_r \times_2 B_r \times_3 C_r$, contributes to the overall tensor T.

Flexibility and Adaptability:

- Having an index r allows for flexibility. The model can adapt and use different combinations of D, A, B, and C for each r.
- This adaptability is crucial in capturing diverse patterns and relationships within the data.

Reduction of Parameters:

- By breaking down T into R terms, the model becomes more parameter-efficient.
- Each r-th term involves its set of matrices, reducing the overall number of parameters compared to dealing with a single large tensor T.

Structured Decomposition:

- The use of r makes the decomposition structured and organized. Each r-th term represents a specific contribution to the overall model.
- This structured approach simplifies the mathematical representation and facilitates better understanding and op ↓ zation.

2. Structural Constraint:

$$T = D_{bd} \times_1 A \times_2 B \times_3 C$$

This equation introduces a structural constraint that allows for a simpler representation. D_{bd} is a block-superdiagonal tensor that combines all the D matrices into a more organized structure.

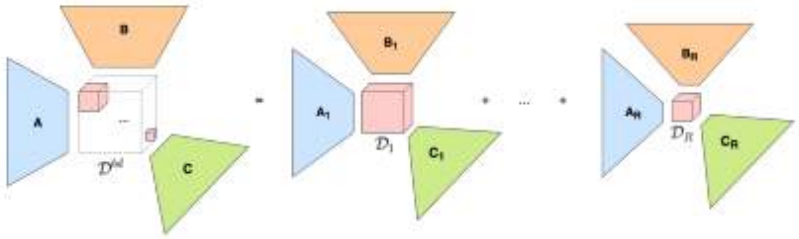


Figure 1: BLOCK framework. The third-order interaction tensor is decomposed in R rank-(L,M,N) terms. We give here the two equivalent representations of the block-term decomposition, presented in this paper. On the left, we show the formulation with the block-superdiagonal tensor decomposition that corresponds to Eq. (4). On the right, we express it as a sum of small decompositions, as written in Eq. (3). Through the R number of blocks and the dimensions of the $\boldsymbol{A}, \boldsymbol{B}$ and $\boldsymbol{C}$ projections, we can handle the trade-off between model complexity and expressivity.

3. Expressing y with respect to x1 and x2:

$$y = T \times_1 x_1 \times_2 x_2$$

This equation shows how the output y is obtained by combining T with the input vectors x1 and x2.

4. Projections:

$$\hat{x}_1 = Ax_1, \hat{x}_2 = Bx_2$$

These equations represent projections of x1 and x2 using matrices A and B.

HTML Content

This equation means that each element of the projected vector $\hat{x}_1$ is obtained by combining the elements of x_1 with the corresponding row of matrix A. It's like transforming x_1 into a new representation that captures certain aspects emphasized by the matrix A

HTML Content

So, in simpler terms, think of x_1 and x_2 as your original data, and the matrices A and B act like filters or lenses. The projections $\hat{x}_1$ and $\hat{x}_2$ represent the data in a way that highlights certain features or patterns, as determined by these matrices. It's a way of focusing on specific aspects of the input data that are important for the subsequent steps in the model

5. **Fusion with Block-Superdiagonal Tensor D_{bd}:**

   $z_r = D_r \times_1 \hat{x}_{1_{rL:(r+1)L}} \times_2 \hat{x}_{2_{rM:(r+1)M}}$

   This equation merges the projections $\hat{x}_1$ and $\hat{x}_2$ using the block-superdiagonal tenso D_{bd} to produce vectors z_r.
6. **Concatenation of z_r Vectors:**

   $z = [z_1, z_2, ..., z_R]$

   This equation combines all the z_r vectors into a single vector z.
7. **Final Prediction:**

   $y = Cz$

   Finally, the prediction vector y is obtained by multiplying the concatenated vector z with matrix C.

Linear Model:In the context of this statement, a linear model involves a matrix. This matrix is responsible for mapping input features to output predictions. Each element in the matrix represents a weight that the model learns during training.
The "hypothesis space" refers to all the possible functions or relationships that the model can learn. In the case of a linear model, it's all the different ways the input features can be combined to produce predictions.
The rank of a matrix is a measure of its "dimensionality" or the number of independent columns (or rows) it has. In the context of linear models, constraining the rank means limiting the number of independent patterns the model can learn from the data.
By constraining the rank, you effectively reduce the number of parameters in the model. Fewer parameters mean a simpler model that is less prone to overfitting and requires less data to learn.
But for Bi linear Model we have the following:
The idea is to use block-term tensor decomposition as a fully learnable set of parameters.