

Data Science

Lecture 11: Multimodal Data Processing

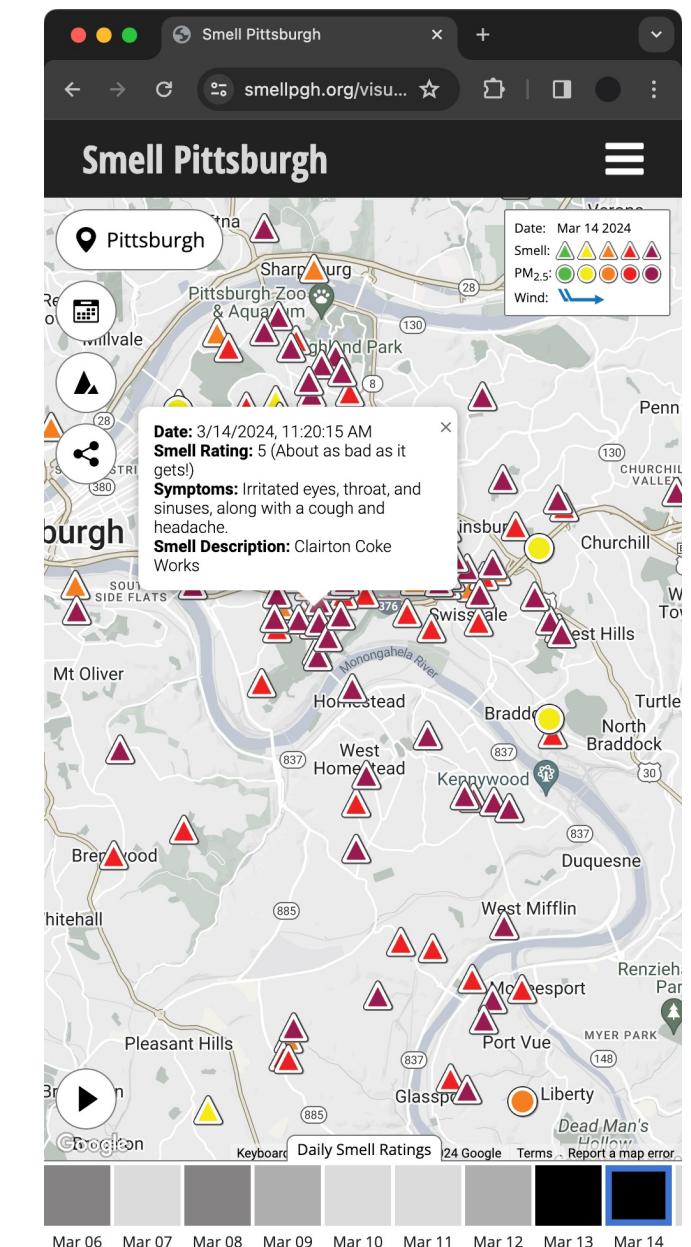
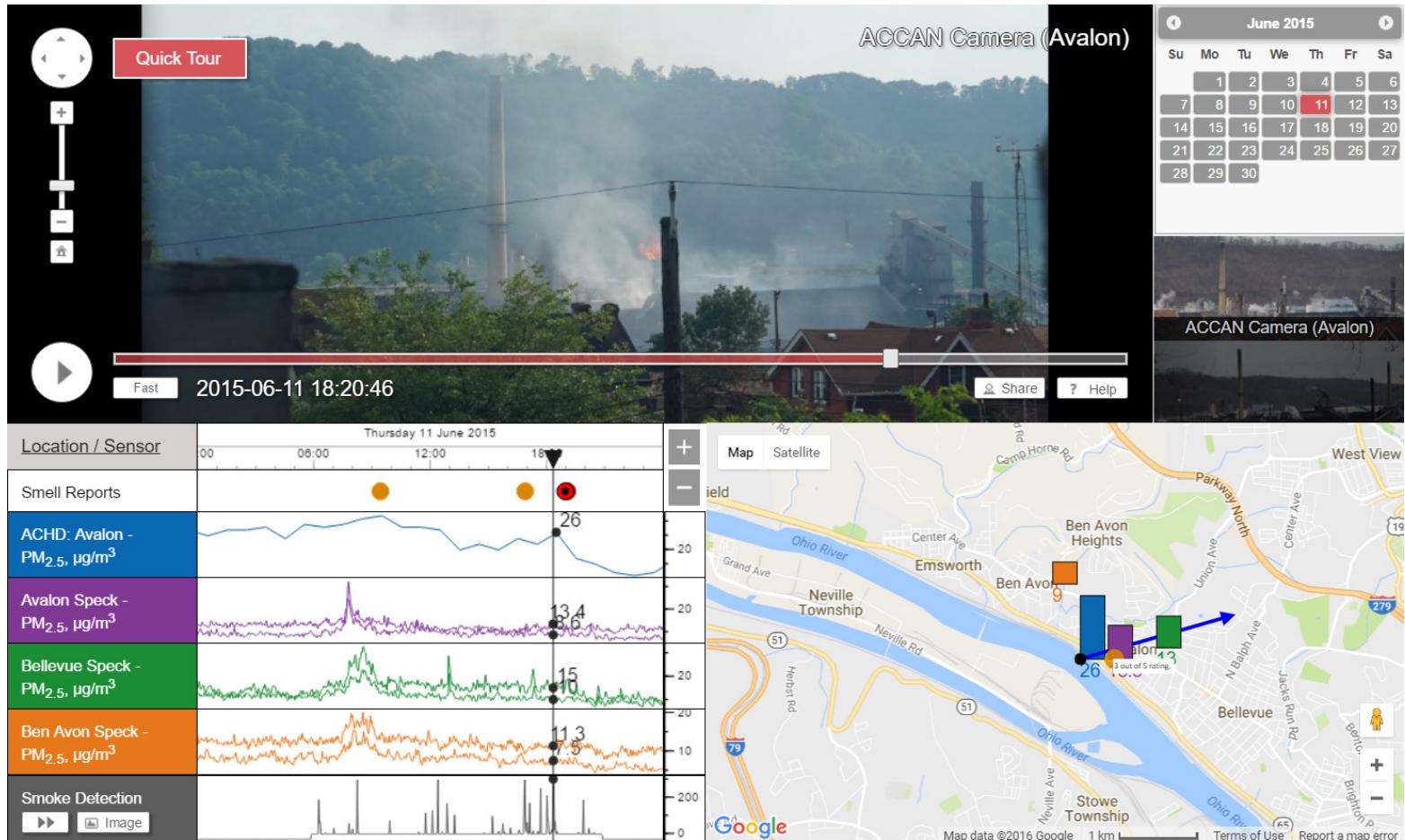


UNIVERSITY
OF AMSTERDAM

Lecturer: Yen-Chia Hsu

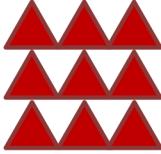
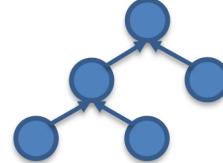
Date: Mar 2026

A **modality** means how a natural phenomenon is perceived or expressed. **Multimodal** means having multiple modalities.



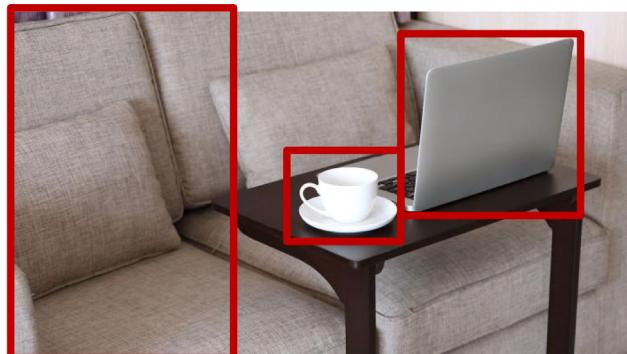
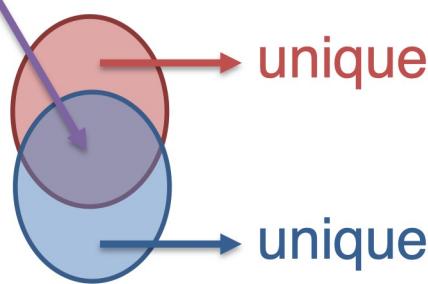
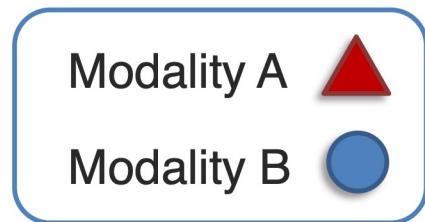
Different modalities can have different characteristics.

- 1 **Element representations:**
Discrete, continuous, granularity
- 2 **Element distributions:**
Density, frequency
- 3 **Structure:**
Temporal, spatial, latent, explicit
- 4 **Information:**
Abstraction, entropy
- 5 **Noise:**
Uncertainty, noise, missing data
- 6 **Relevance:**
Task, context dependence

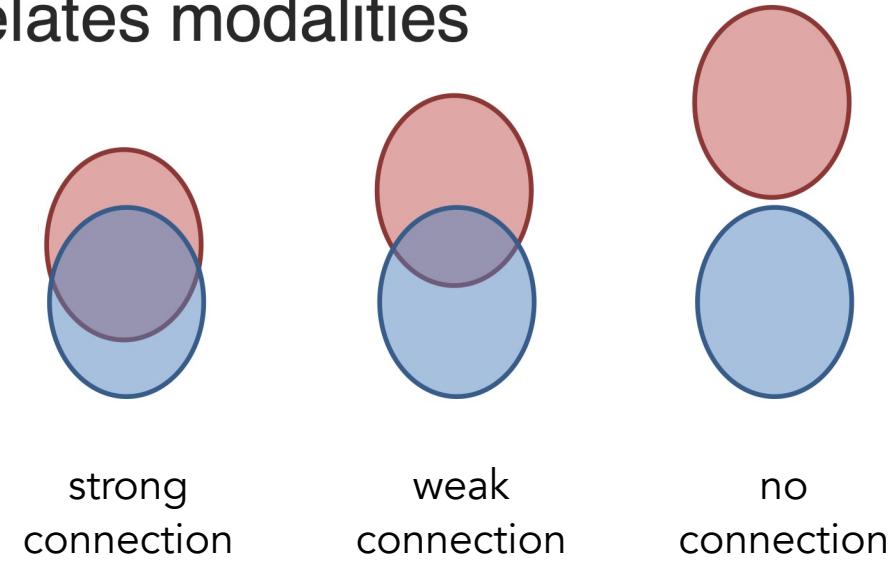
Modality A (e.g., vision)	Modality B (e.g., language)	Examples (e.g., vision/language)
		<ul style="list-style-type: none">Sets of imagesSets of characters
		<ul style="list-style-type: none">Pixel densityWord frequency
		<ul style="list-style-type: none">Nearby objectsSemantics
$H(\Delta)$	$H(\bullet)$	<ul style="list-style-type: none">Ordering of wordsMotion, color
		<ul style="list-style-type: none">OcclusionAmbiguity
		<ul style="list-style-type: none">Object detectionMachine translation

Different modalities can share information with different levels of connections.

Connected: Shared information that relates modalities



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



The shared information can be connected in different ways.

Association



e.g., correlation,
co-occurrence

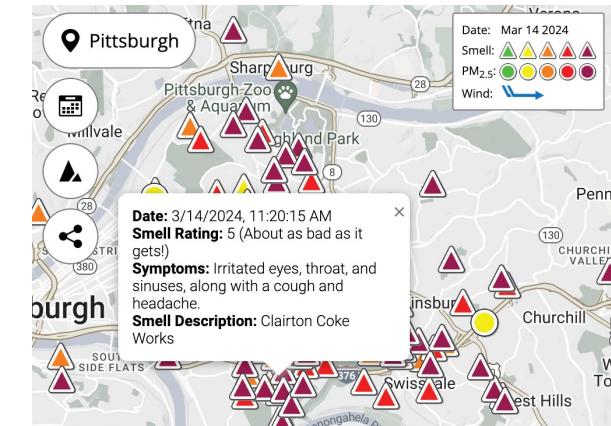


A **teacup** on the **right** of a **laptop**
in a **clean room**.

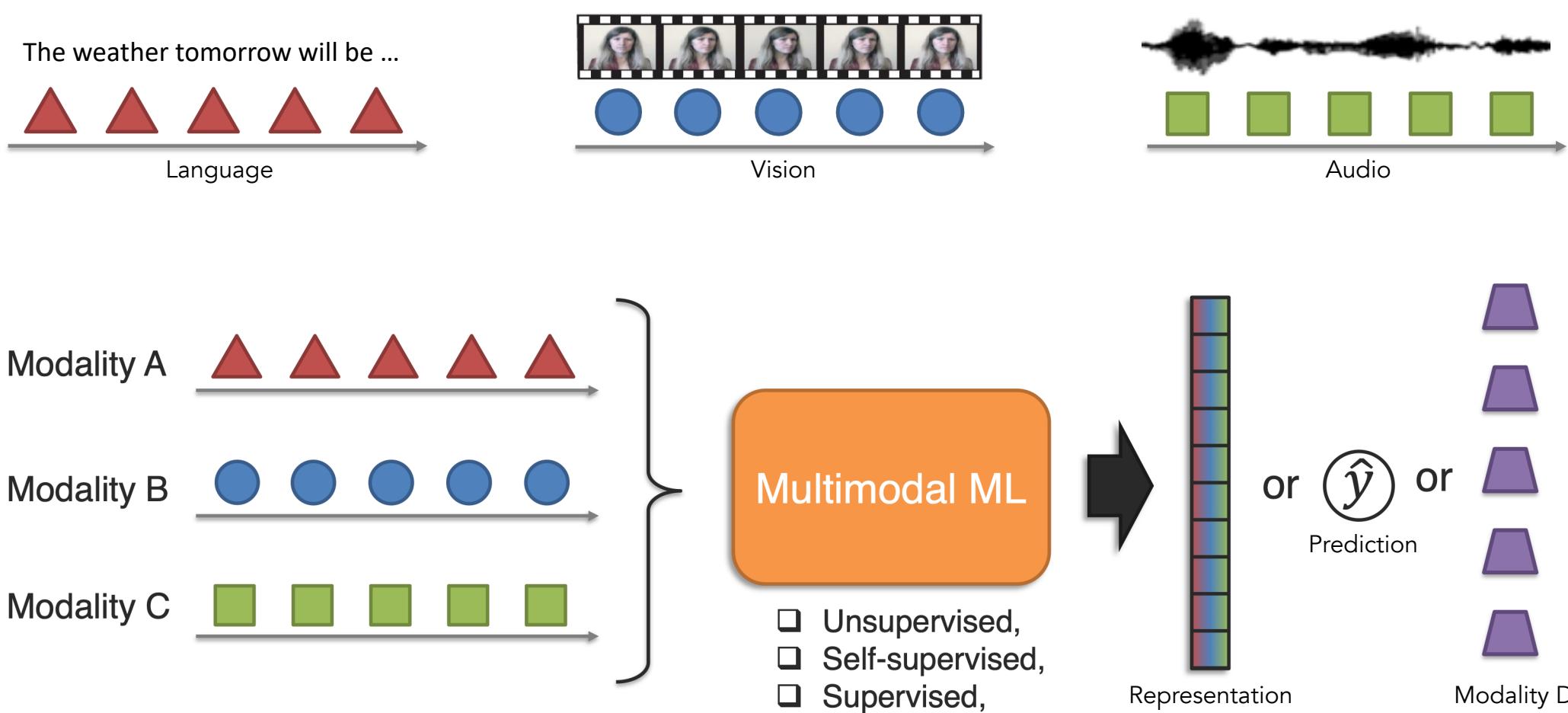
Dependency

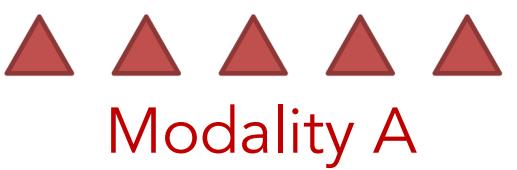


e.g., causal,
temporal

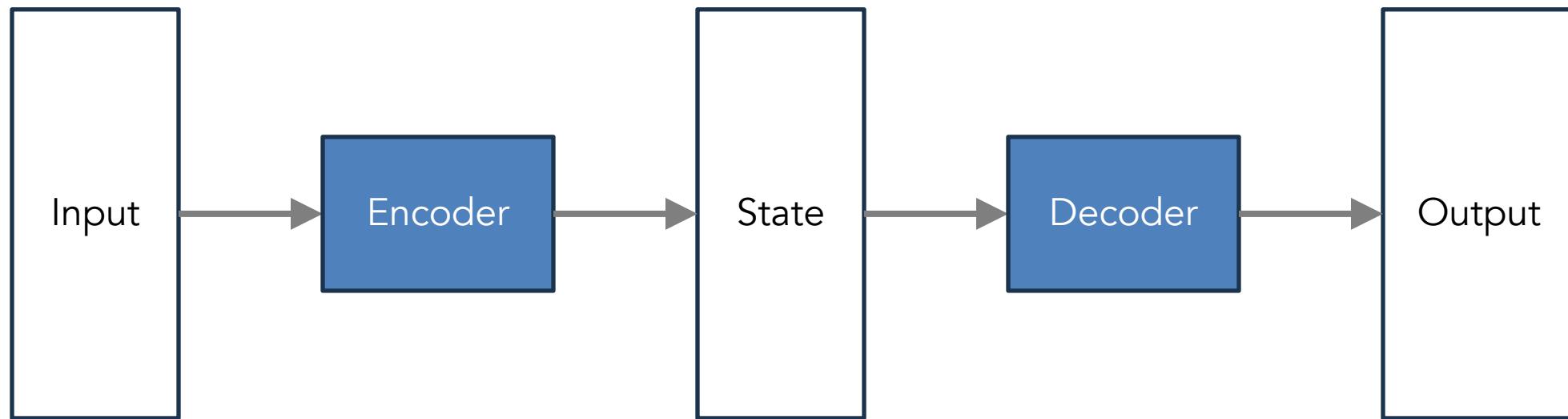


Multiple modalities can exist in different parts of the machine learning pipeline.

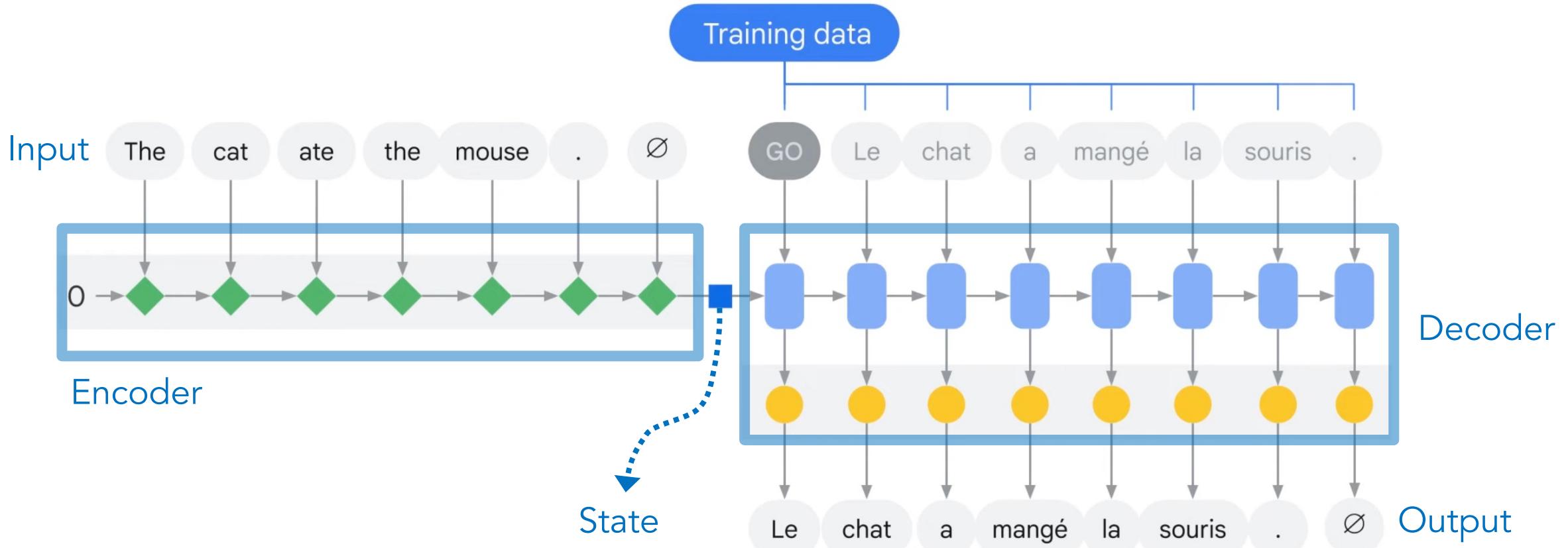




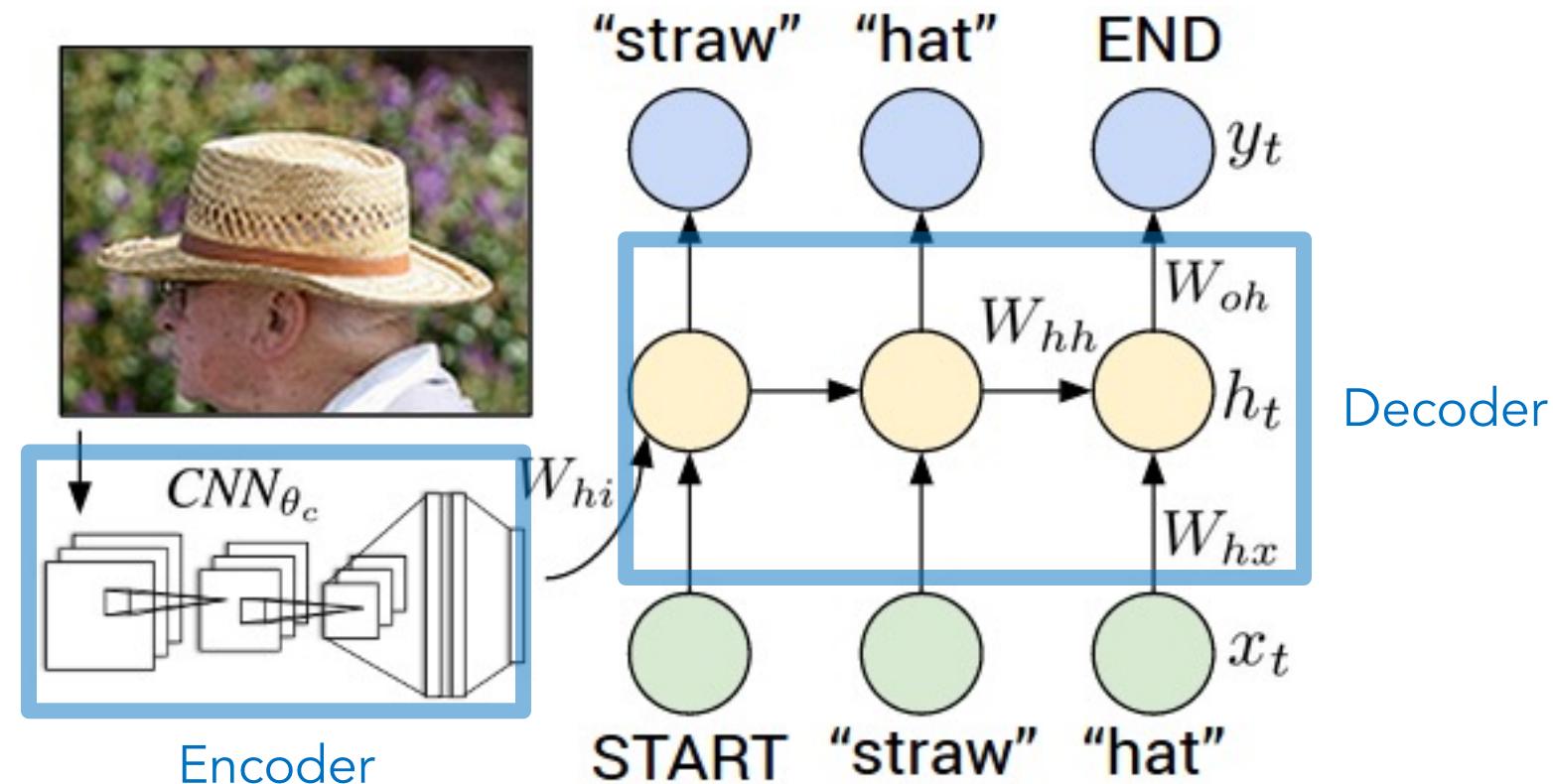
From a high-level point of view, we can encode inputs (e.g., a sequence of language tokens, image with pixels, or patches of images with pixels) into an intermediate state and then decode the intermediate state into the outputs that are suitable for our task.



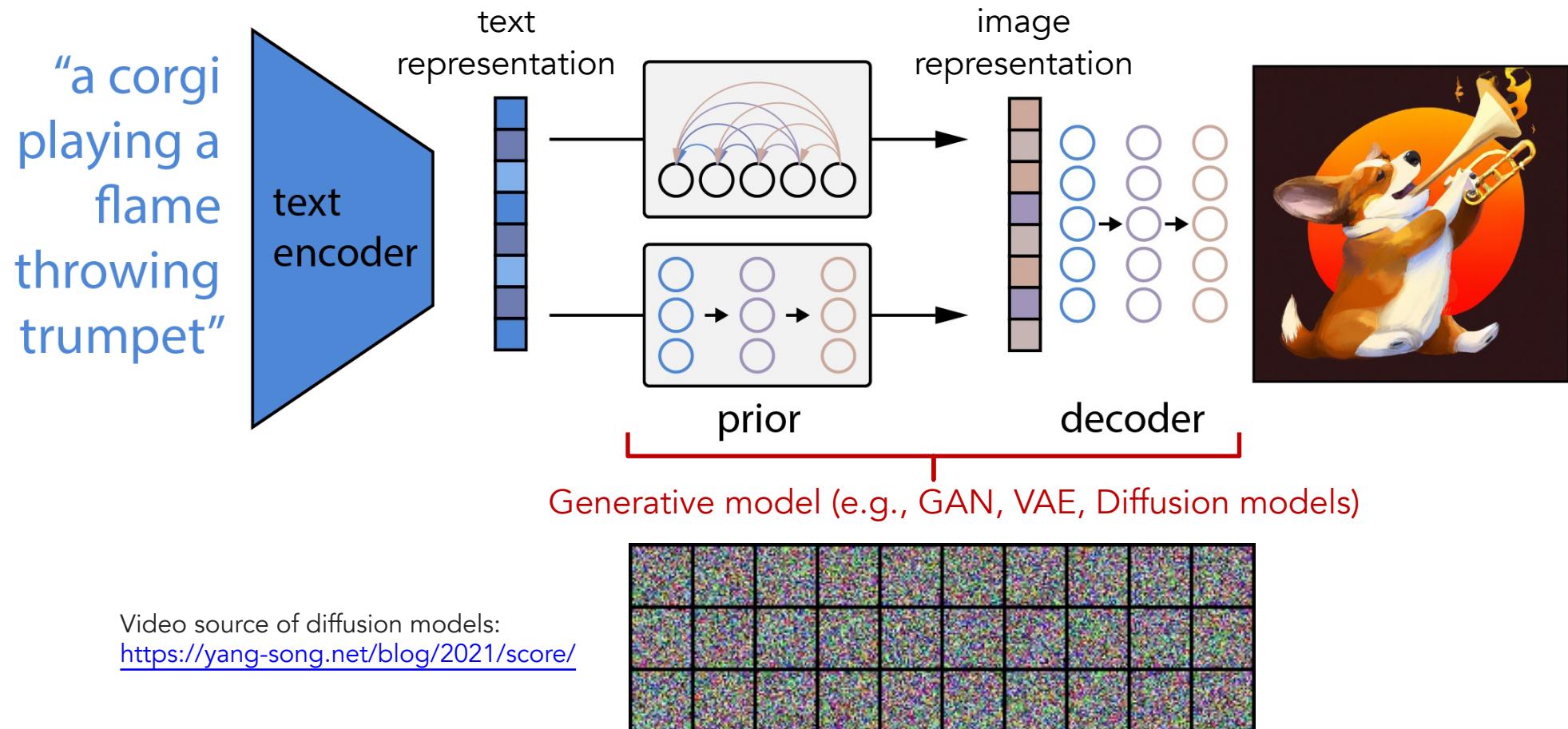
In the single modality setting, we can encode the entire input sequence (e.g., a sentence) into an intermediate state and then decode the state into another sequence.



In the multimodal setting, such as Image Captioning, the model takes images as input and then outputs sentences that describe the input images (**vision→language**).



We can also take text as input and then generate images that match the input text (language→vision).

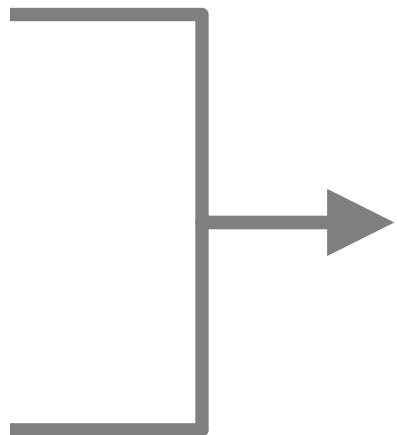




Modality A



Modality B



\hat{y}

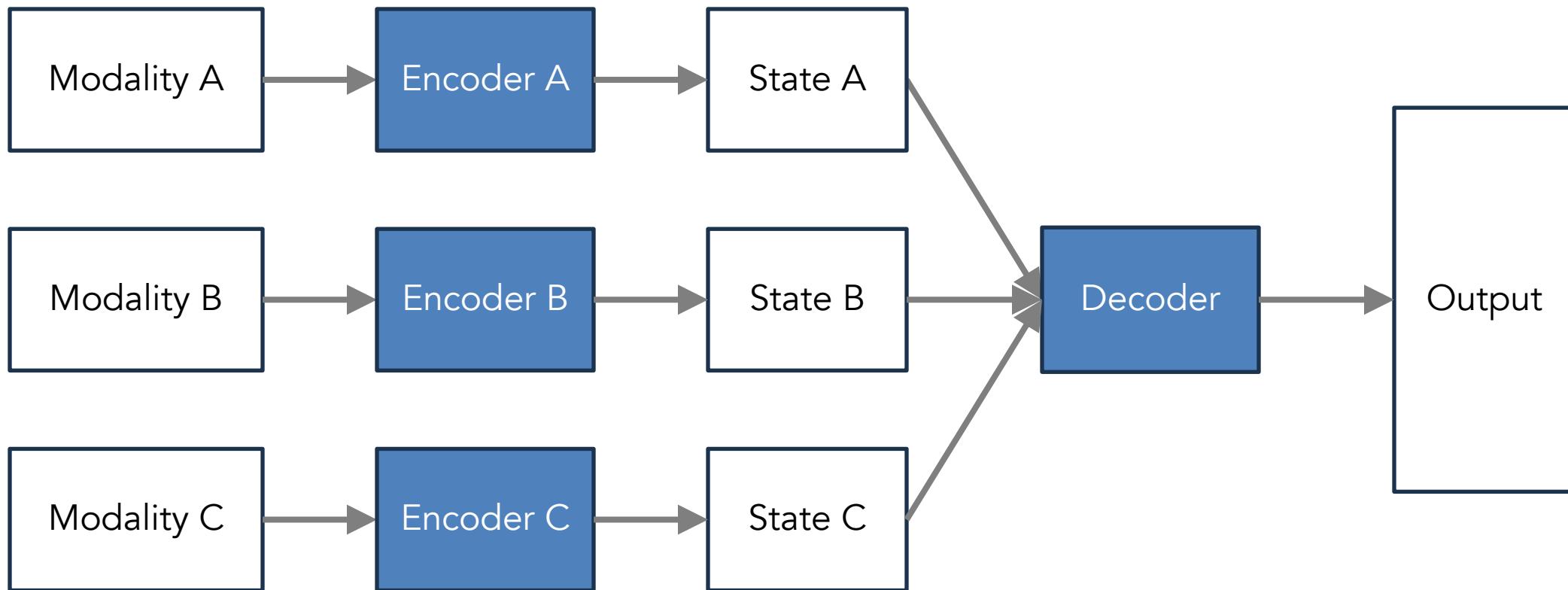
Prediction

OR

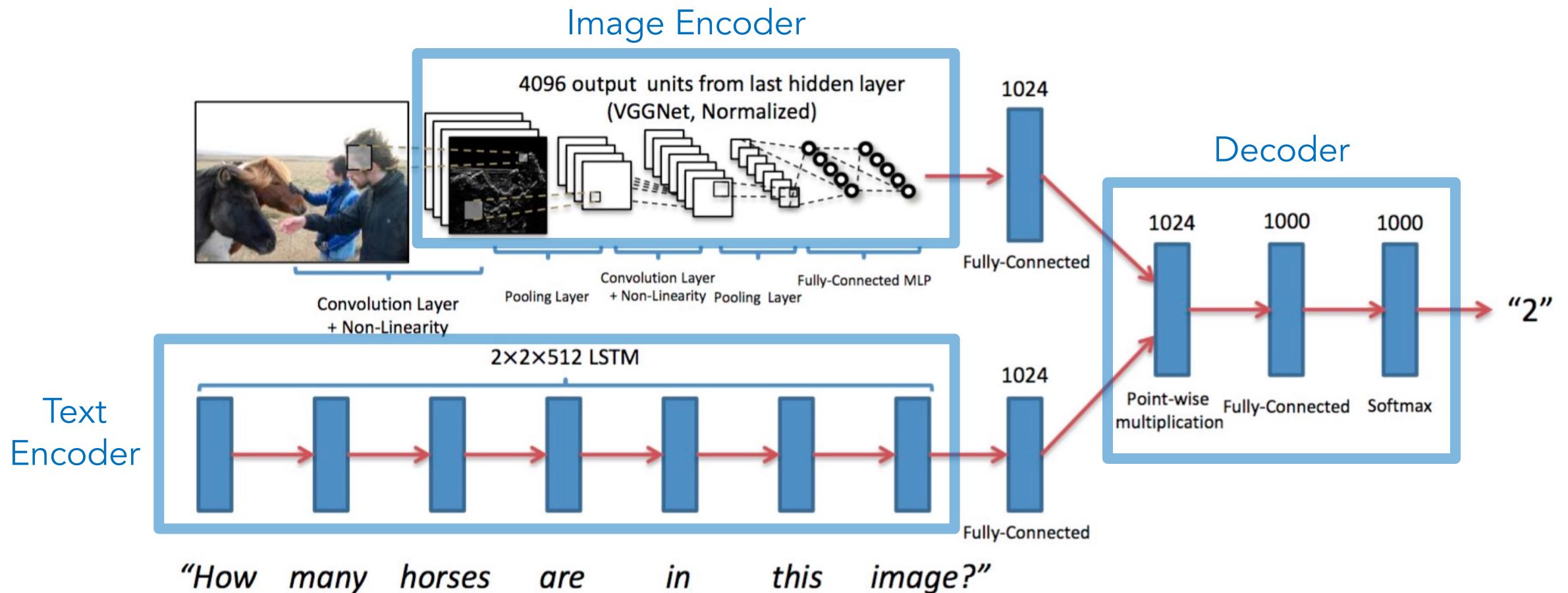


Modality C

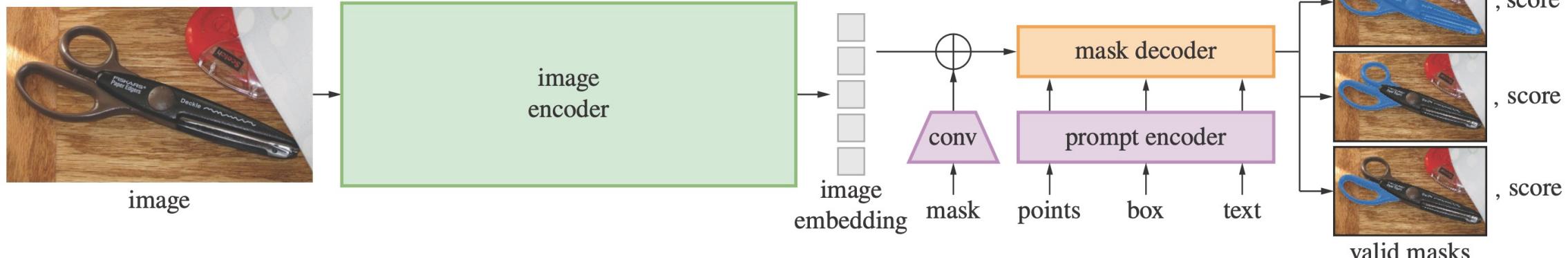
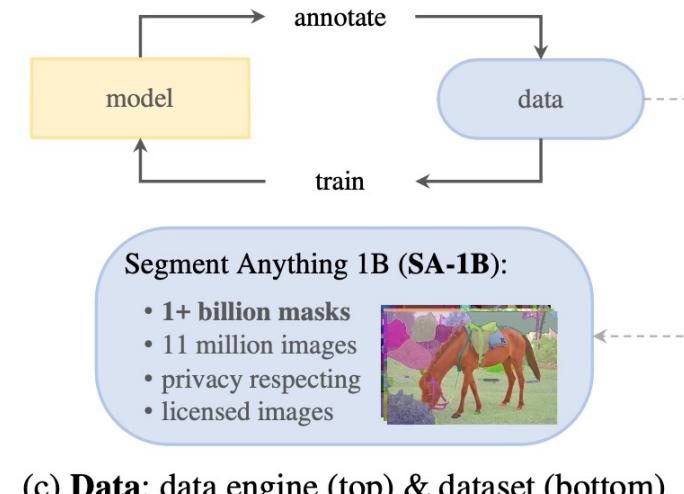
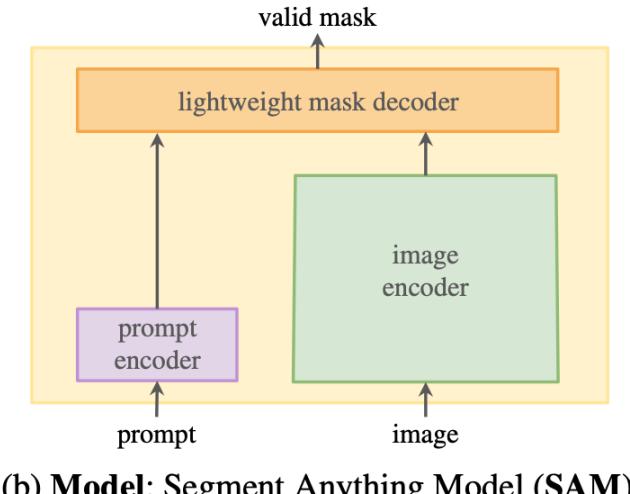
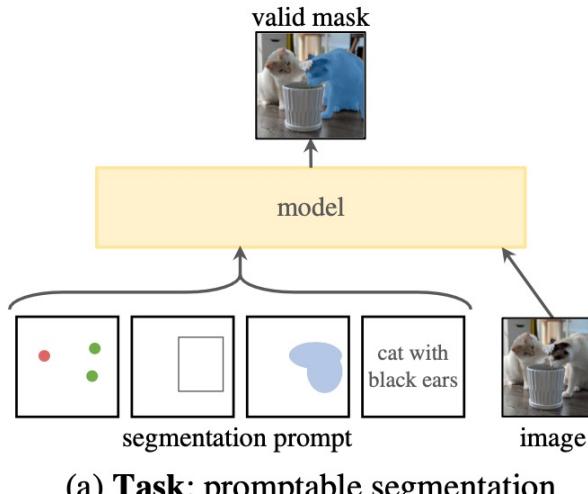
Applying the encoder-decoder concept, one approach is to encode each modality individually and put all the encoded states into the decoder to produce outputs.



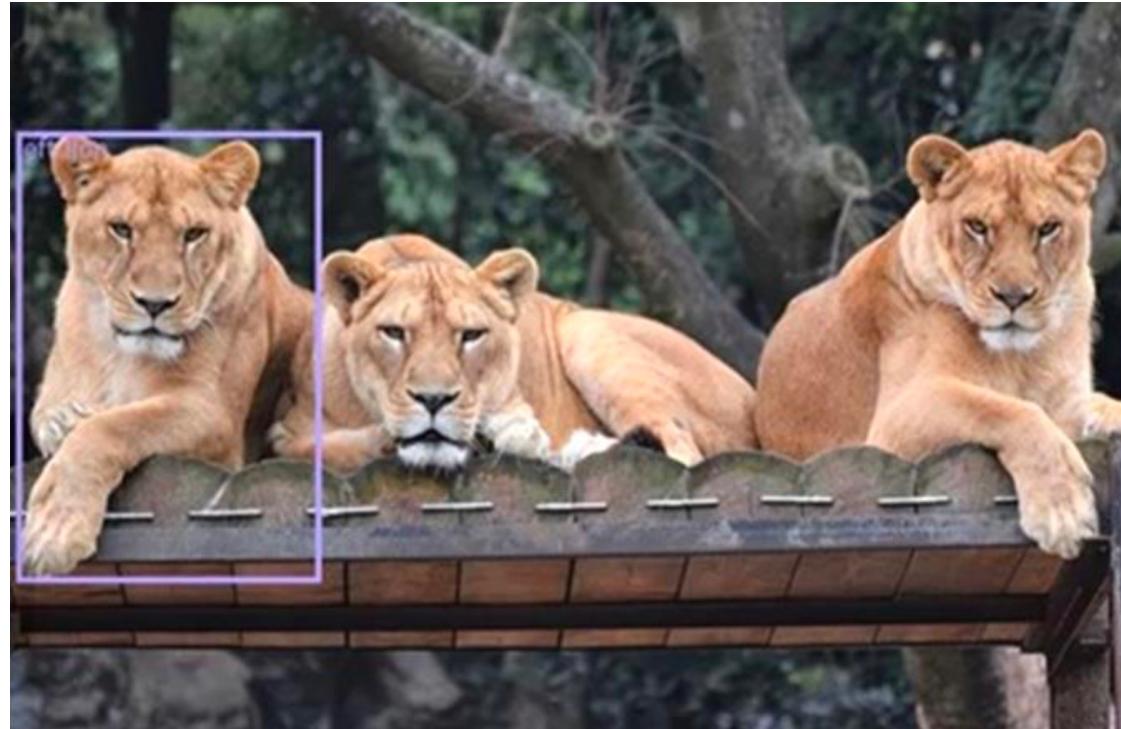
Visual Question Answering takes both images and sentences as input and then outputs a label of text-based multiple-choice answer (**vision+language→label**).



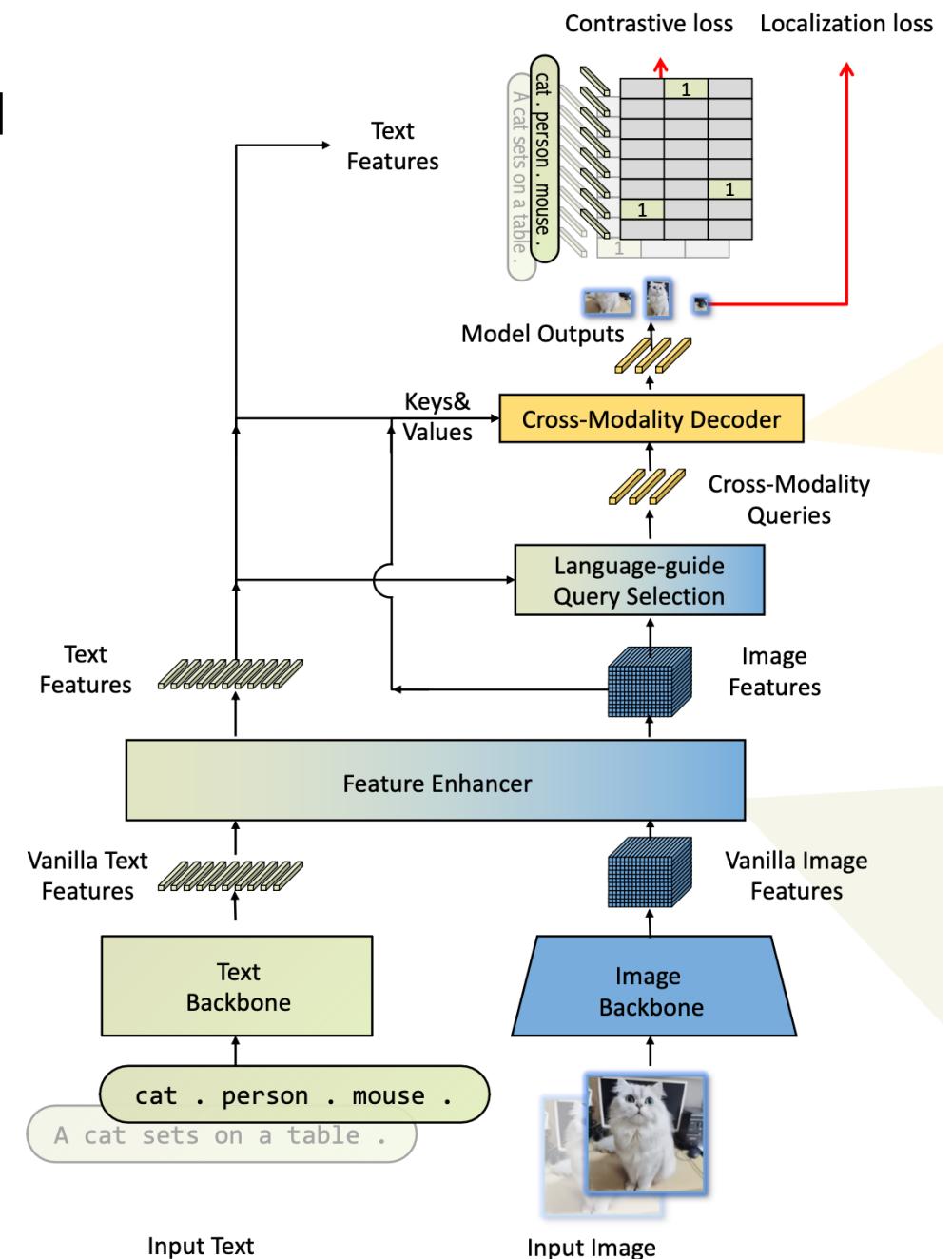
The Segment Anything model takes images and prompts as input and then outputs a pixel-level mask of different segmented regions (**vision+prompts→mask**).



Grounding DINO takes images and text as input and then outputs a bounding box (**vision+text→box**).



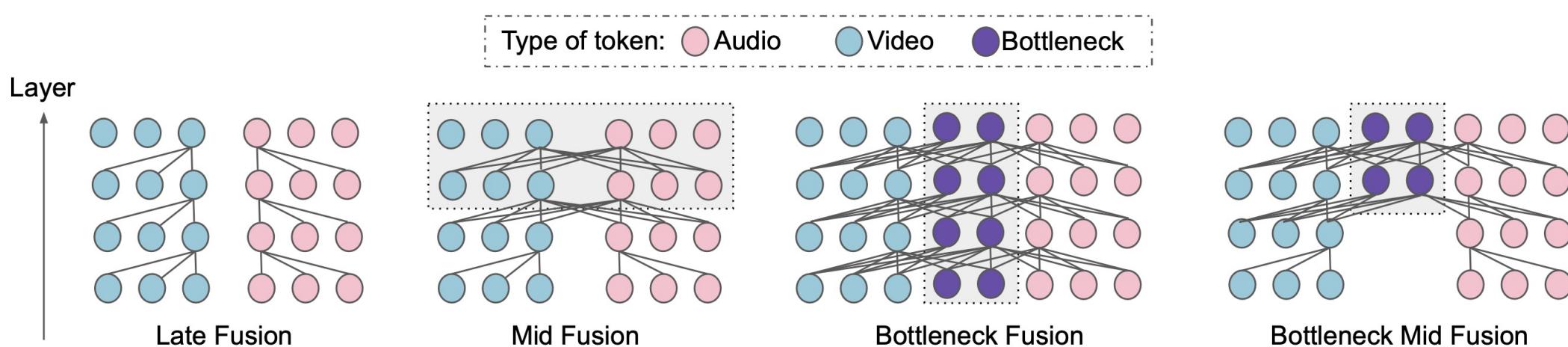
Text prompt: "The left lion"



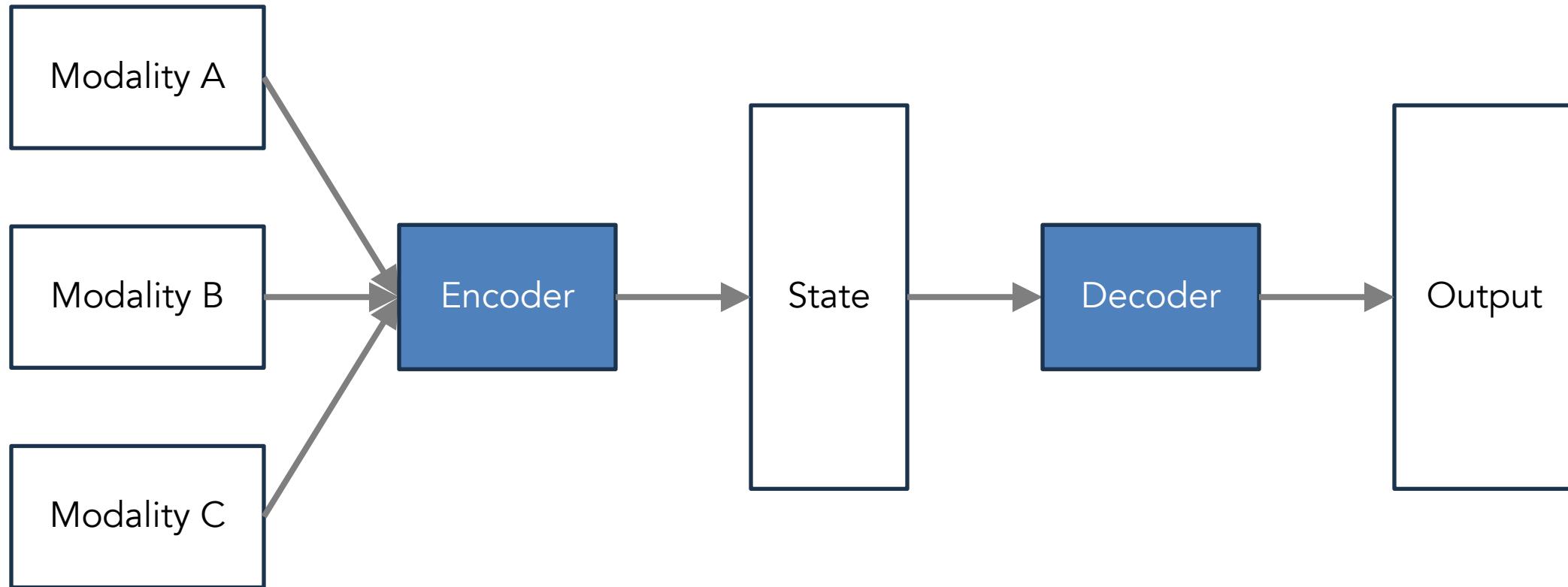
When the input has multiple modalities, we can **fuse the modalities** or explicitly learn their connections in the model architecture.



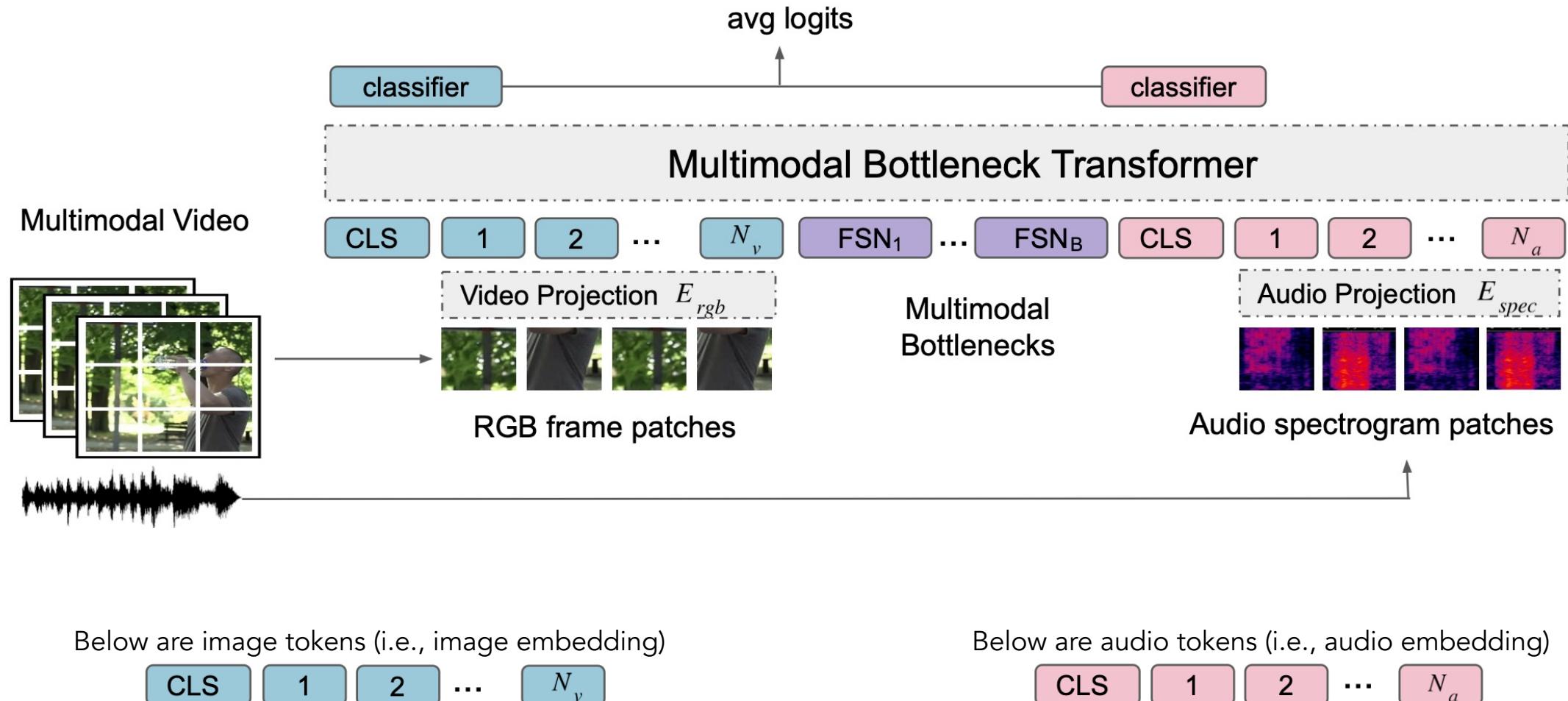
Source: CMU 11-777 MMML Course Lecture 1.1 -- <https://cmu-multicomp-lab.github.io/mmml-course/fall2023/schedule/>



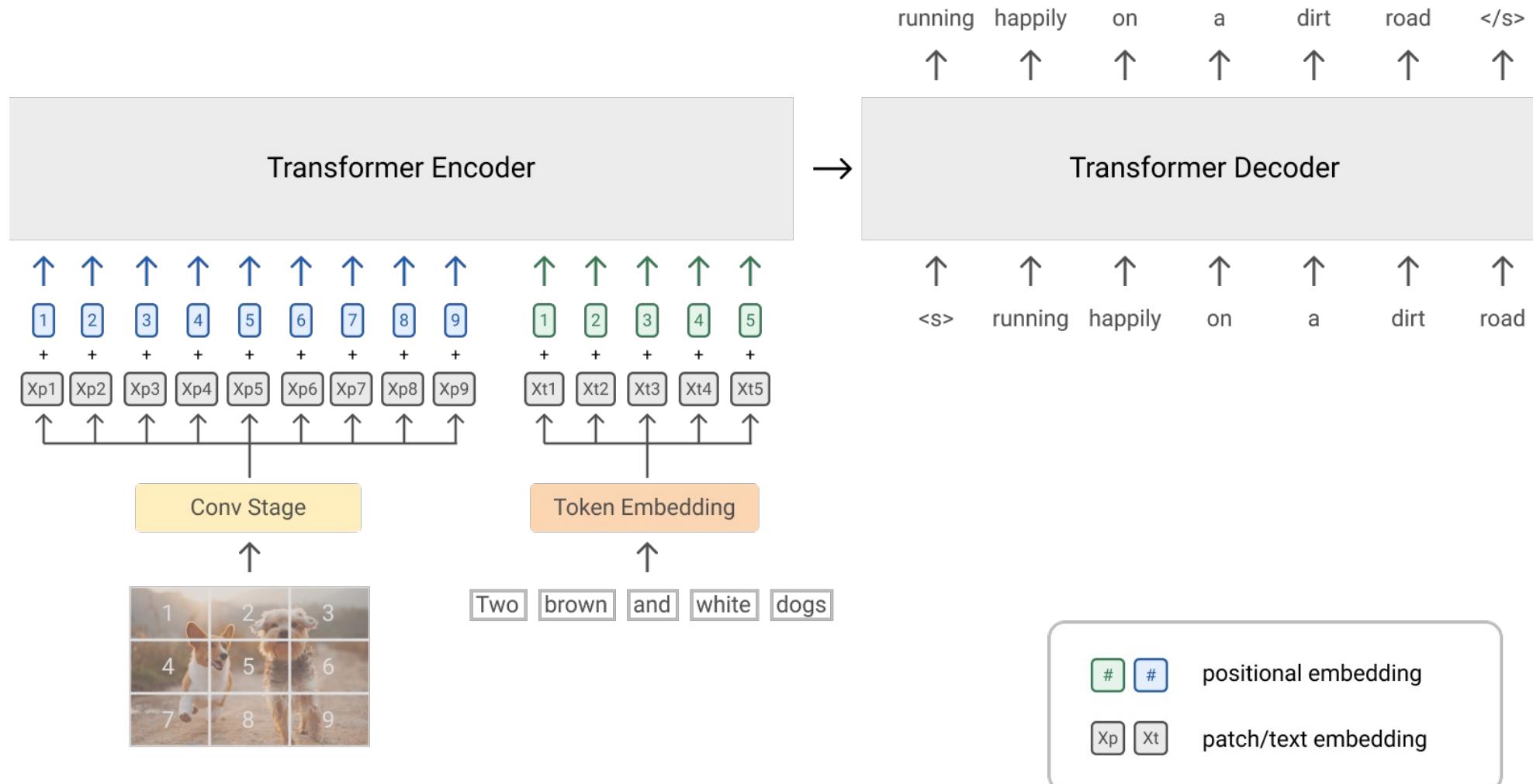
Instead of using multiple encoders, we can also encode all modalities into one state and then put the state into the decoder, such as the Transformer architecture.



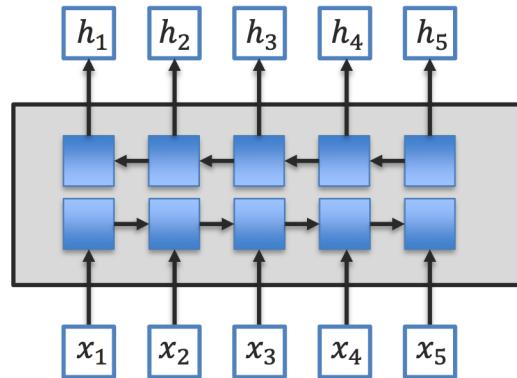
Transformers can use both video and audio signals to predict output categories, which is the Video Classification task (**vision+audio→labels**).



Transformers also work for multiple vision-language tasks (**vision+language** \rightarrow **language**).

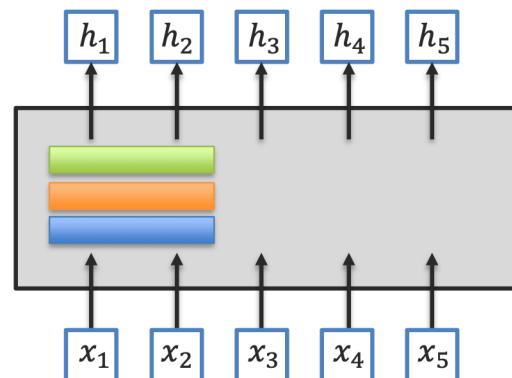


Transformers use **self-attention**, which is a way of encoding sequences to tell how much attention each input should pay attention to the other inputs (including itself).



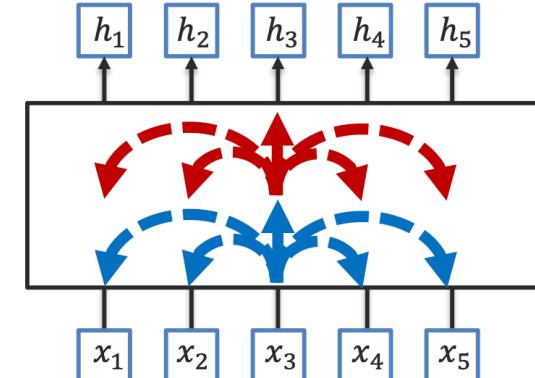
Bidirectional RNN

- Hard to parallelize



Convolution

- Can be parallelized!
- Need multiple layers to model long-range dependencies
- Static convolution kernels



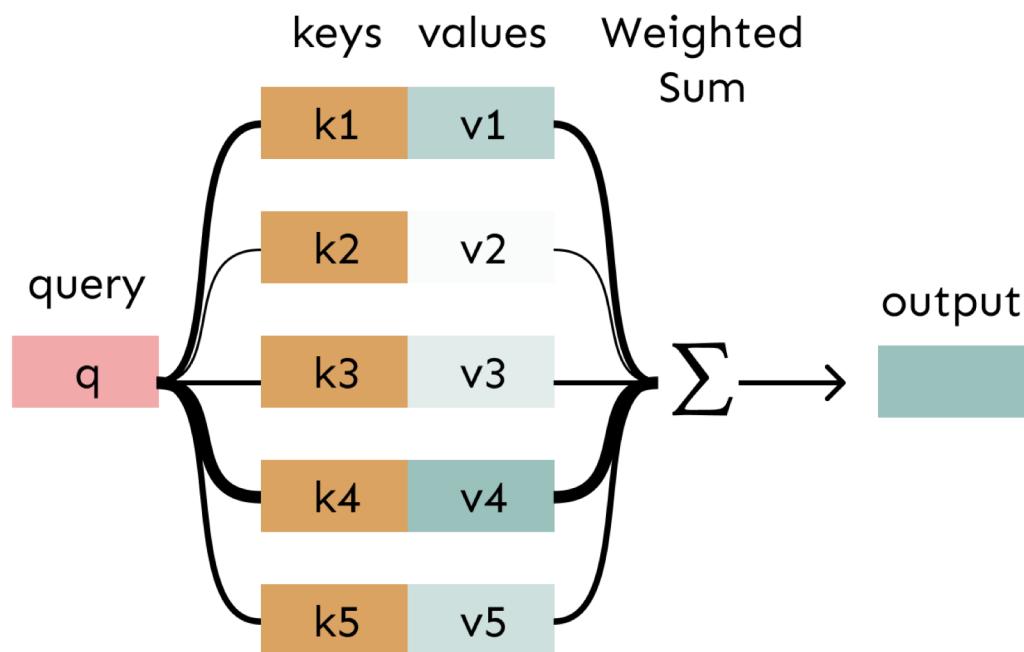
Self-attention

- Can be parallelized!
- Long-range dependencies!
- Dynamic attention weights!

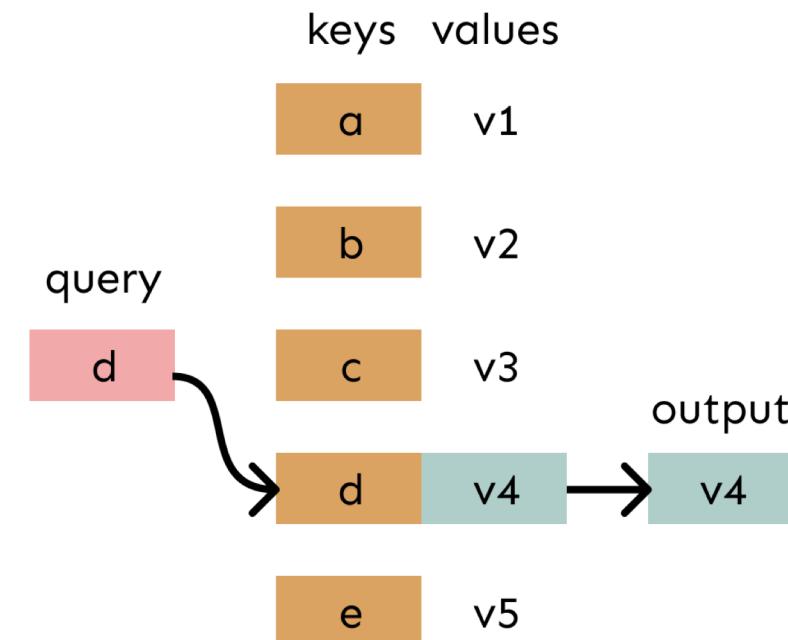
Attention is *weighted* averaging, which lets you do lookups!

Attention is just a **weighted** average – this is very powerful if the weights are learned!

In **attention**, the **query** matches all **keys** *softly*, to a weight between 0 and 1. The keys' **values** are multiplied by the weights and summed.



In a **lookup table**, we have a table of **keys** that map to **values**. The **query** matches one of the keys, returning its value.



Step 5: Compute attention-weighted sum of encoder output:

$$\sum_{t=1}^T a_t v_t$$

Step 4: Compute the attention distribution using softmax:

$$[a_1 \ a_2 \ \dots \ a_T] = \text{softmax}([e_1 \ e_2 \ \dots \ e_T])$$

Step 3: Compute attention scores (dot product similarity):

$$e_t = \text{score}(q, k_t) = q^T k_t$$

W_k is trainable

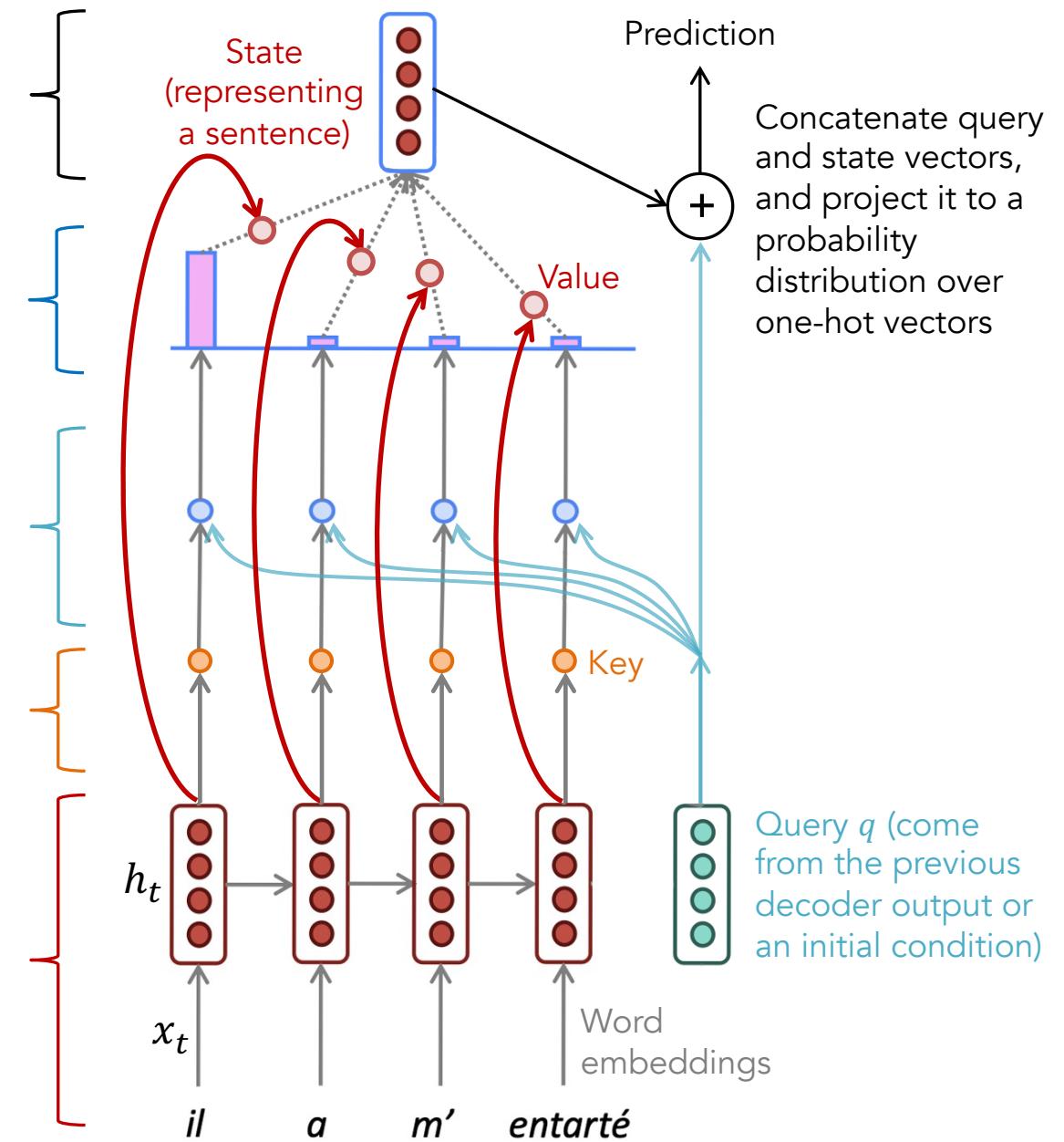
Step 2: Transform encoder outputs (dimension reduction):

$$k_t = \tanh(W_k h_t)$$

Step 1: Get the encoder output values (from the RNN):

$$v_t = h_t = h(x_t)$$

There are many ways of doing step 2 and 3



Step 5: Compute the attention-weighted sum of values:

$$\sum_{t=1}^T a_t v_t$$

Step 4: Compute the attention distribution using softmax:

$$[a_1 \ a_2 \ \dots \ a_T] = \text{softmax}([e_1 \ e_2 \ \dots \ e_T])$$

Step 3: Compute attention scores using a similarity metric:

$$e_t = \text{score}(q, k_t) = q^T k_t$$

W_k is trainable

Step 2: Compute the key vector

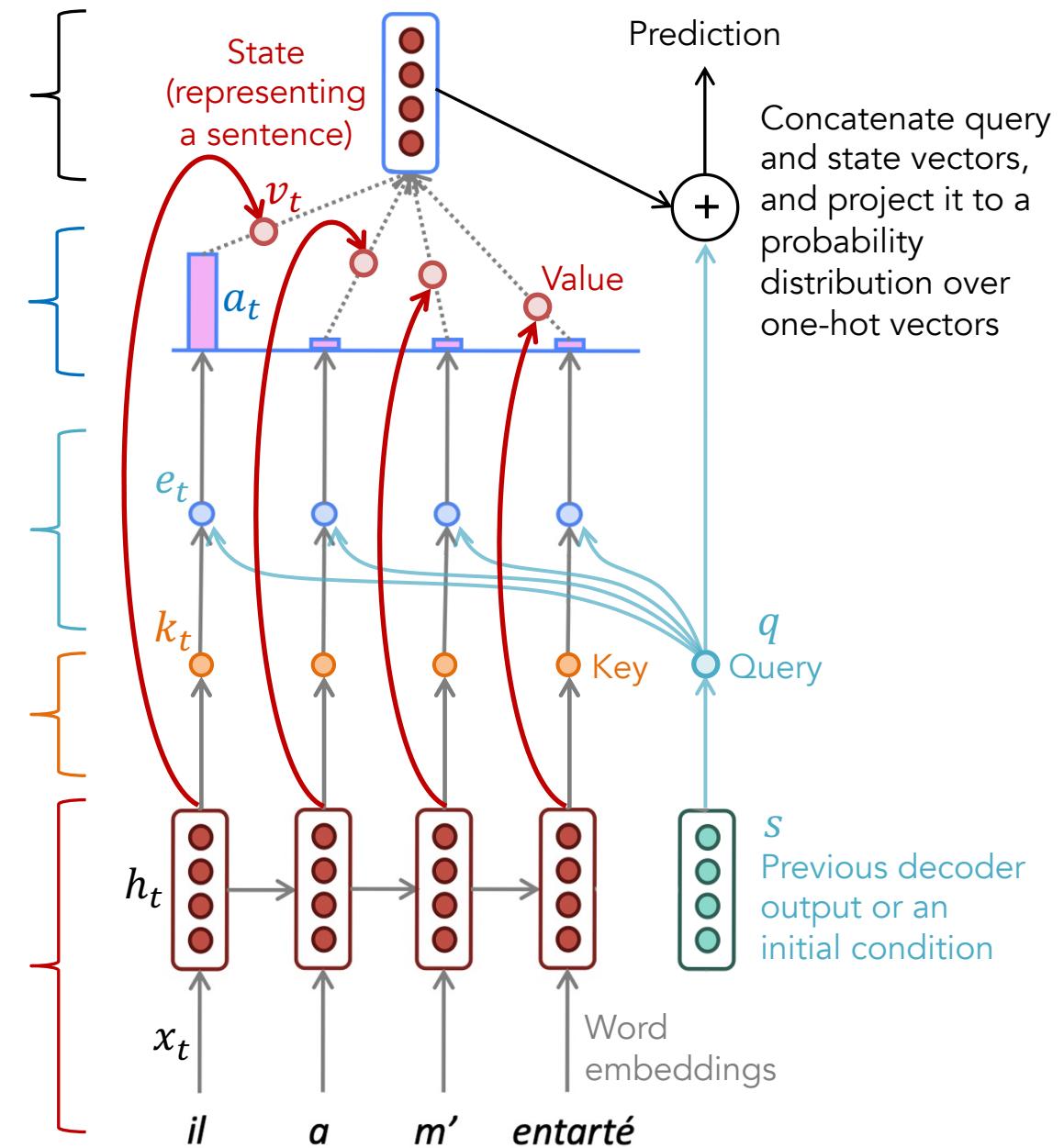
$$k_t = \tanh(W_k h_t)$$

Step 1: Compute the value and query vectors

$$v_t = W_v h_t \quad [\text{but for now } W_v = \mathbf{1}]$$

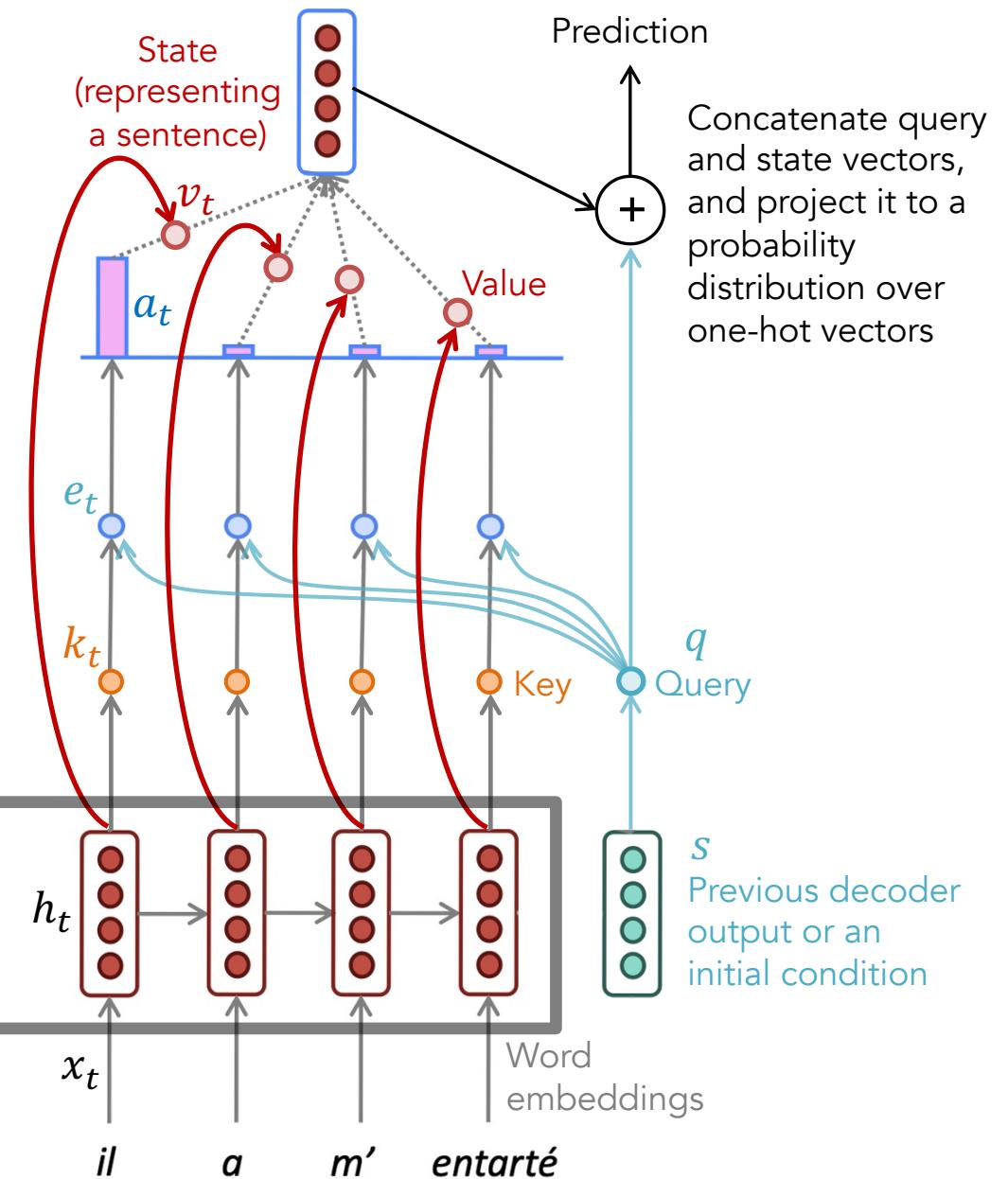
$$q = W_q s \quad [\text{but for now } W_q = \mathbf{1}]$$

(h_t is the encoder output; s is the previous decoder output)



Attention addresses the problem of remembering previous information.
But there is still one problem.

Why is the recurrent neural network structure not desirable?



Step 5: Compute the attention-weighted sum of values:

$$h_t = \sum_{t=1}^T a_t v_t$$

Step 4: Compute the attention distribution using softmax:

$$[a_1 \ a_2 \ \dots \ a_T] = \text{softmax}([e_1 \ e_2 \ \dots \ e_T])$$

Step 3: Compute attention scores using a similarity metric:

$$e_t = \text{score}(q_t, k_t) = (q_t^T k_t) / \sqrt{d_k}$$

(d_k is the dimension of the key and query vectors)

Step 2: Compute the key vector

$$k_t = W_k x_t \quad [\text{there is no tanh function now}]$$

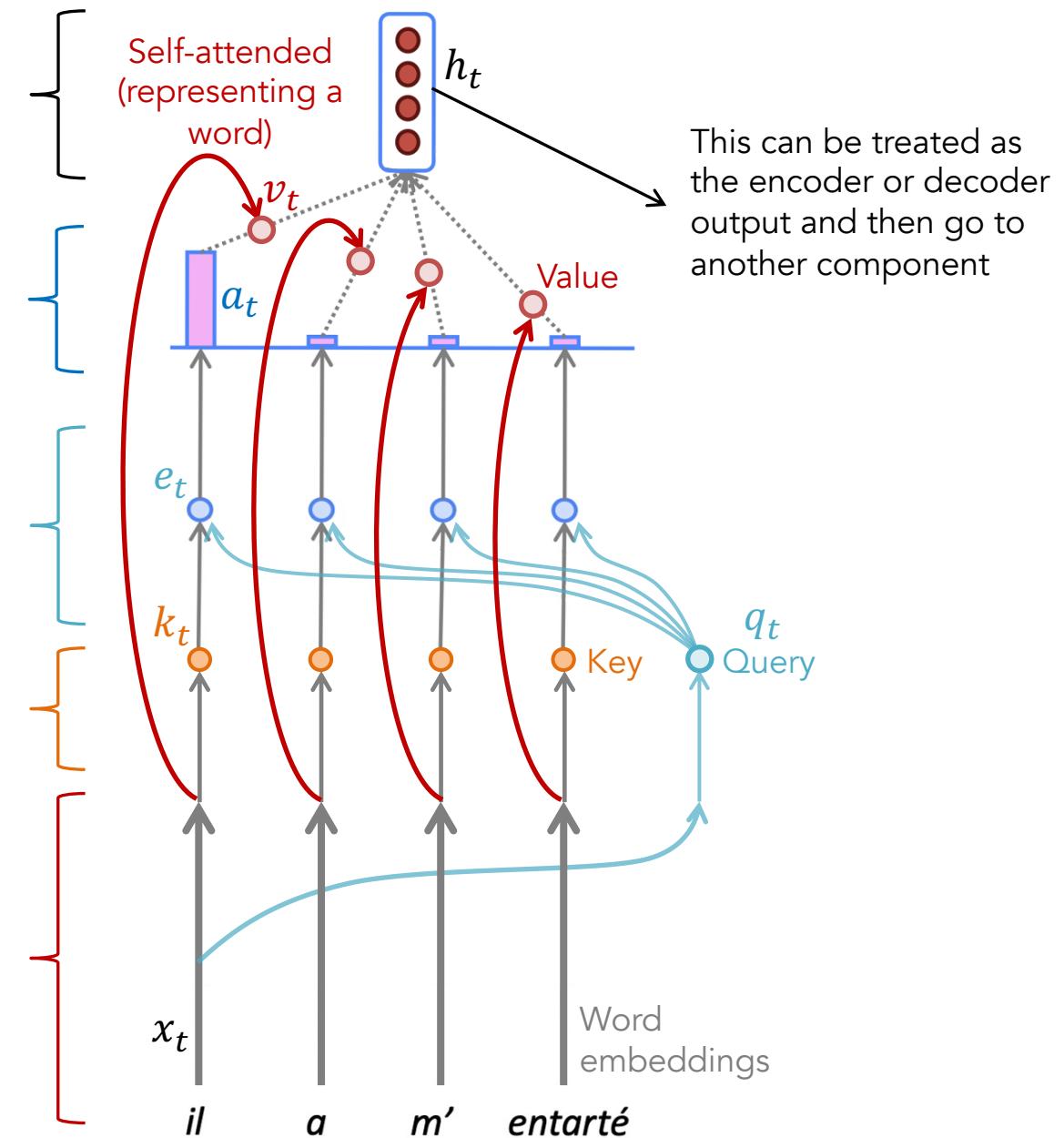
W_q , W_k , and W_v are trainable

Step 1: Compute the value and query vectors

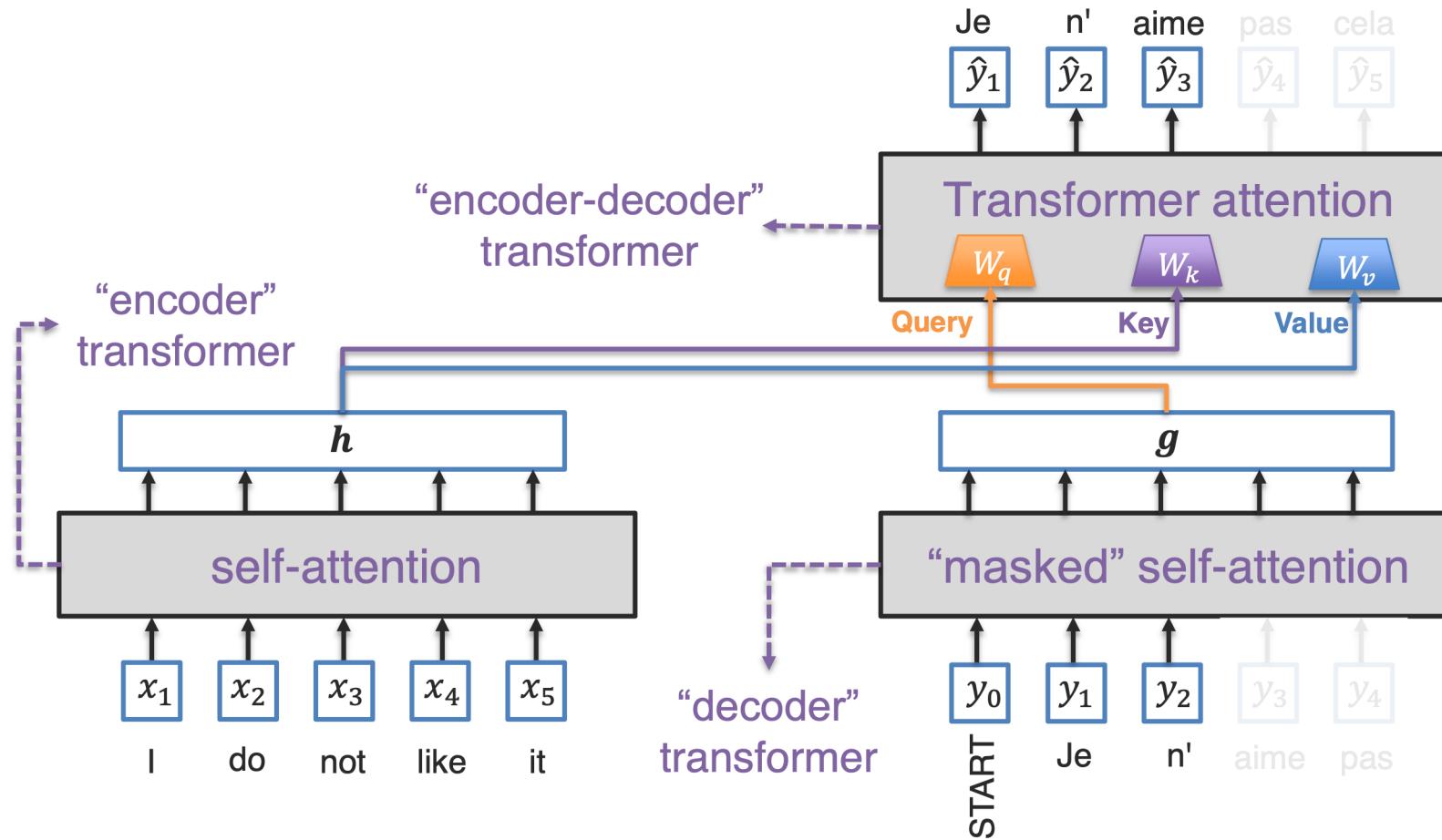
$$v_t = W_v x_t$$

$$q_t = W_q x_t$$

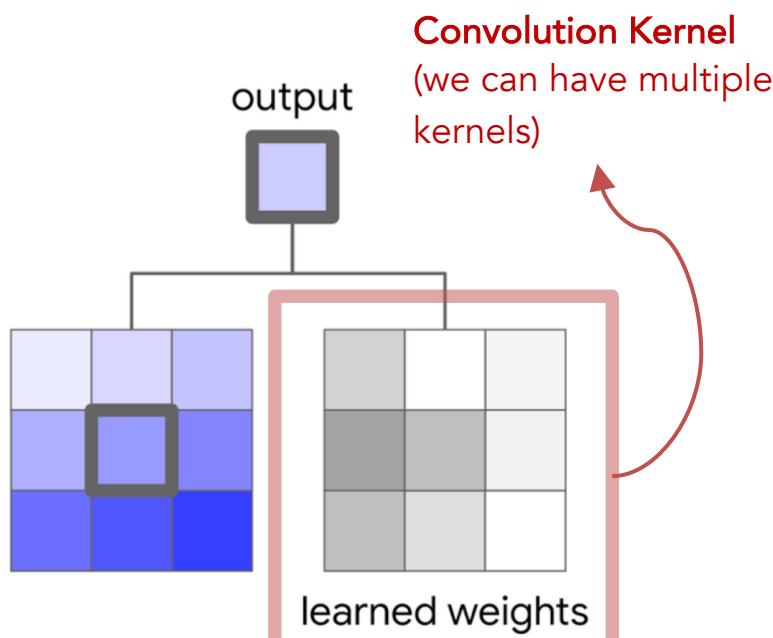
(x_t is the word embedding vector)



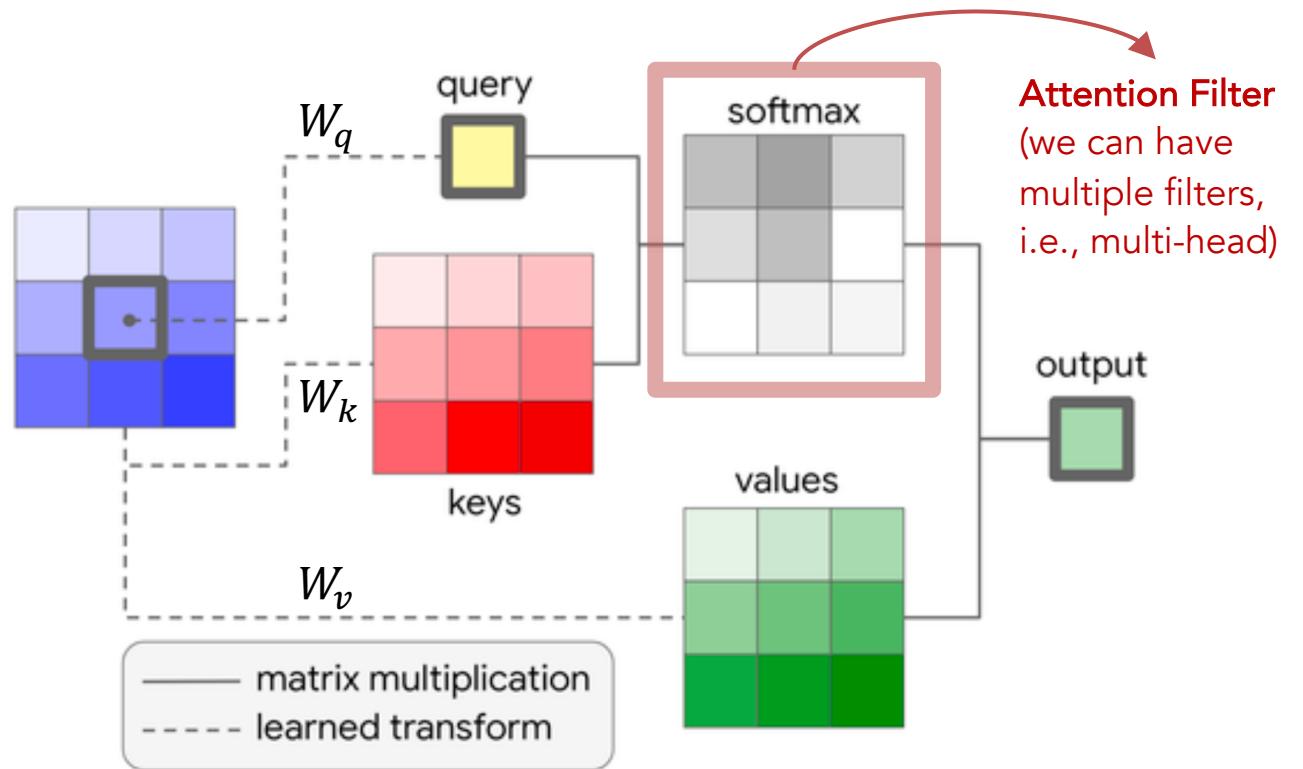
Transformers are connected by two self-attention blocks (one for encoder, one for decoder) and an encoder-decoder attention block (similar to the original attention).



Convolution layers use **fixed weights** (kernels) to filter information. Self-attention layers dynamically compute attention filters to show how well a pixel matches its neighbors.



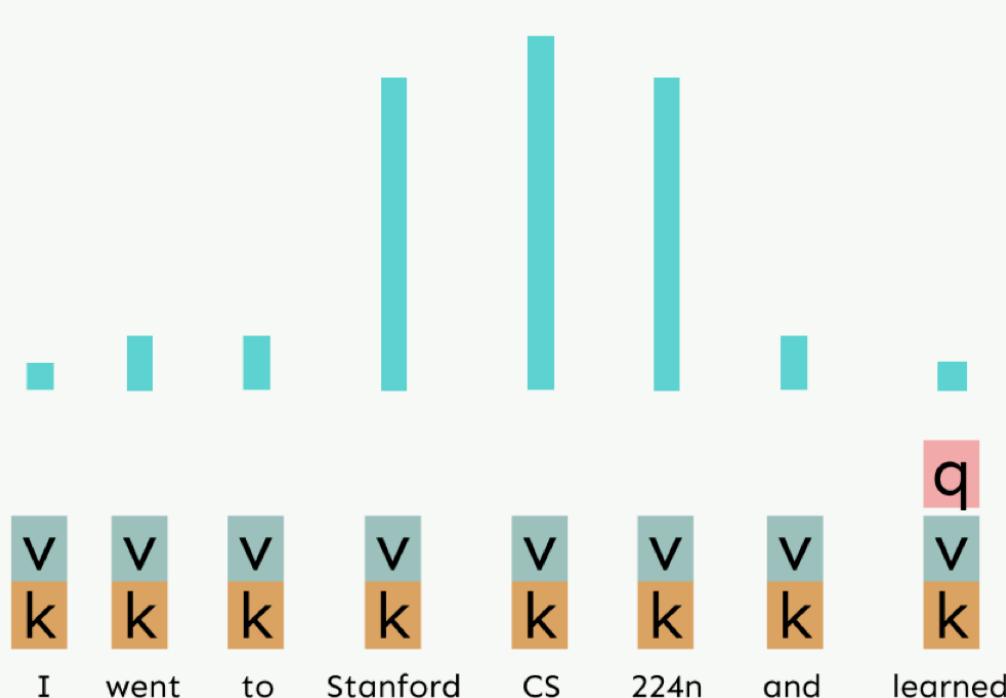
Convolution



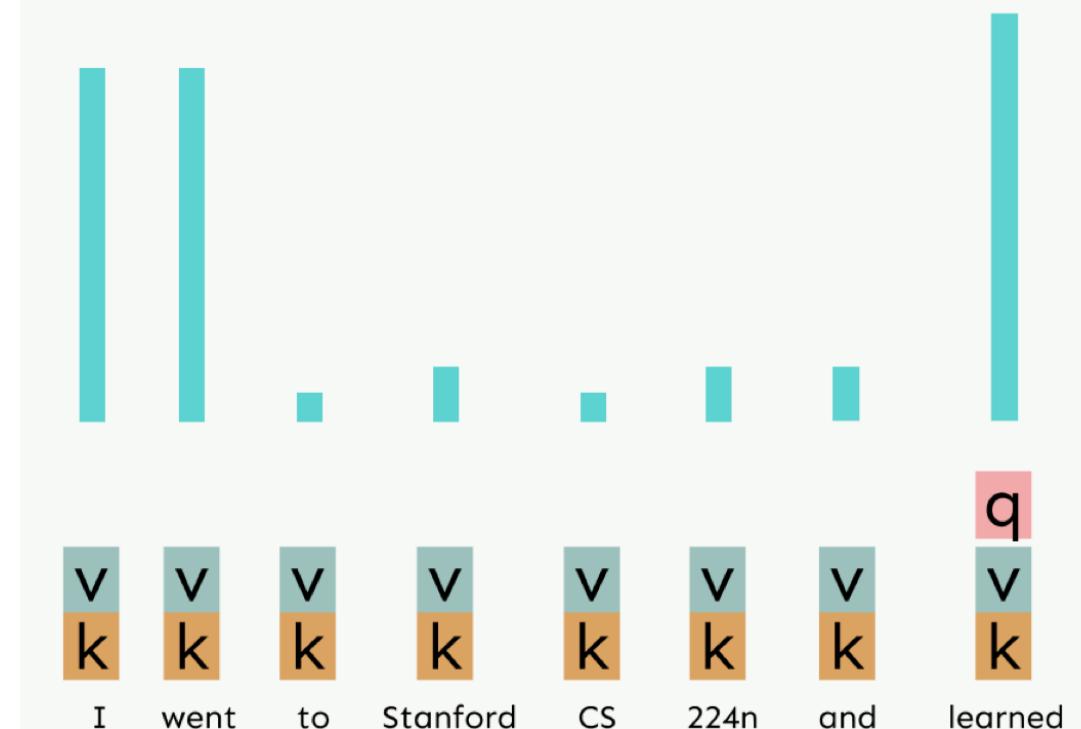
Self-attention

Transformers use **multi-head attention** to look at different aspects of the inputs.

Attention head 1
attends to entities



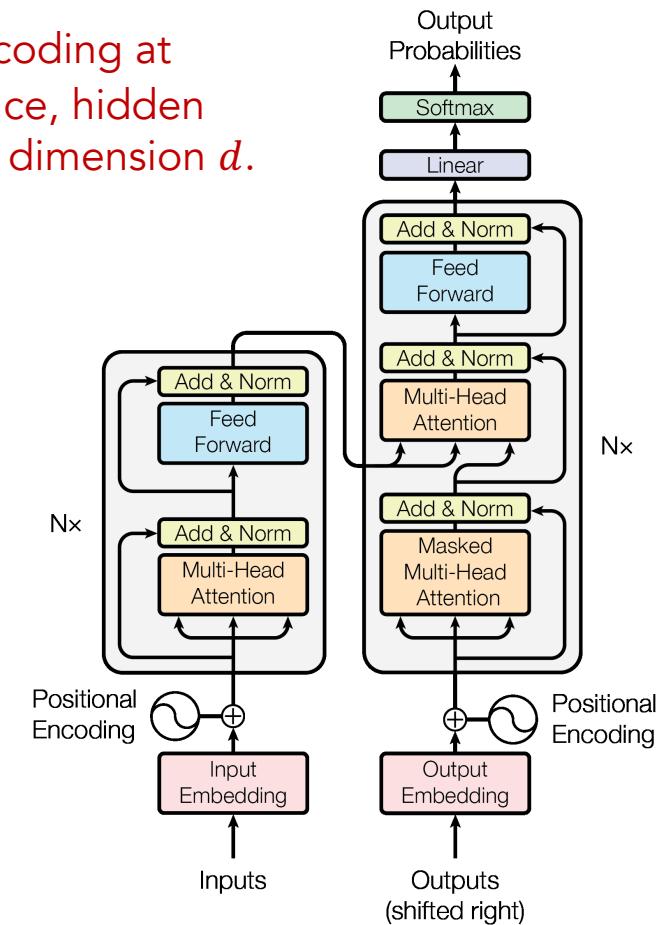
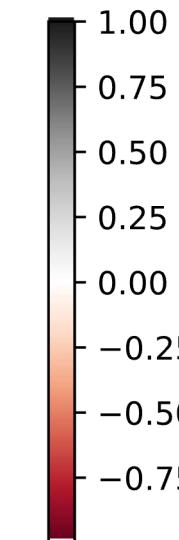
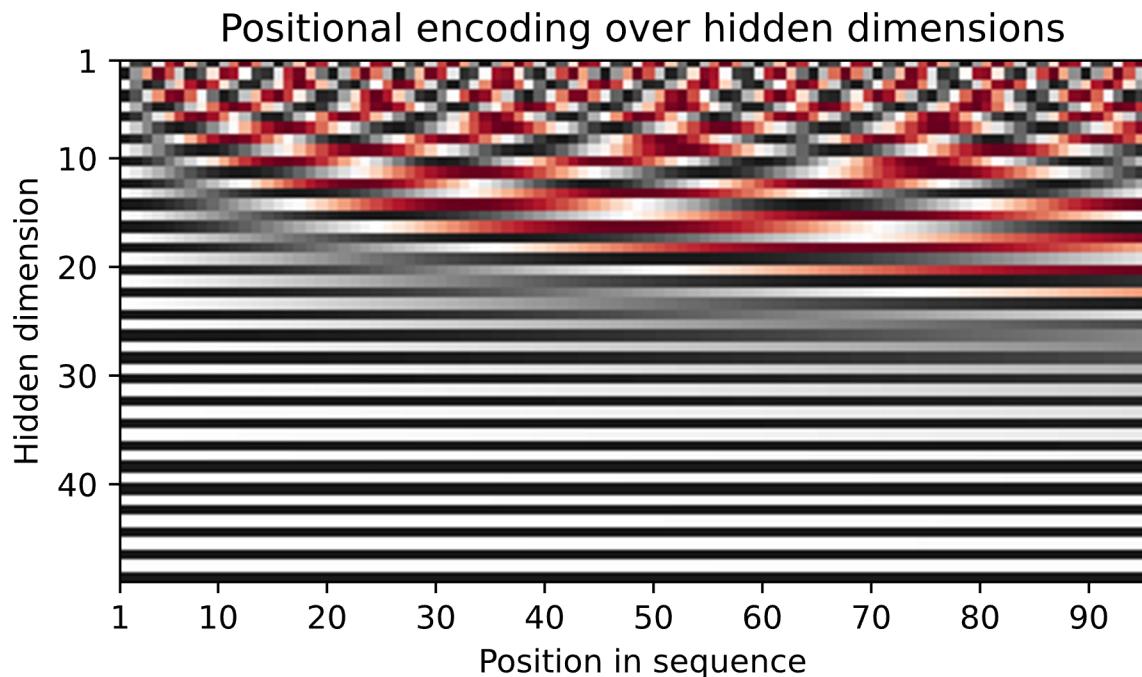
Attention head 2 attends to
syntactically relevant words



Self-attention is permutation invariant (looks at the input as a set of elements). We use **positional encoding** to add the position information to the input embedding vectors.

$$PE_{(pos,i)} = \begin{cases} \sin\left(\frac{pos}{10000^{i/d_{\text{model}}}}\right) & \text{if } i \bmod 2 = 0 \\ \cos\left(\frac{pos}{10000^{(i-1)/d_{\text{model}}}}\right) & \text{otherwise} \end{cases}$$

$PE_{(pos,i)}$ is the position encoding at position pos in the sequence, hidden dimensionality i , and total dimension d .

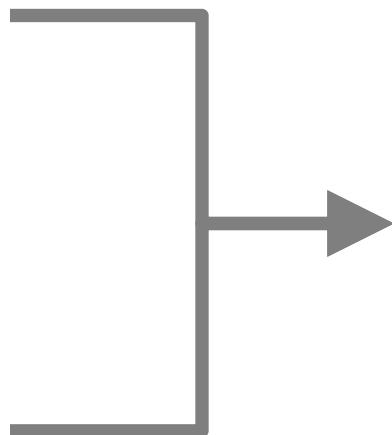




Modality A

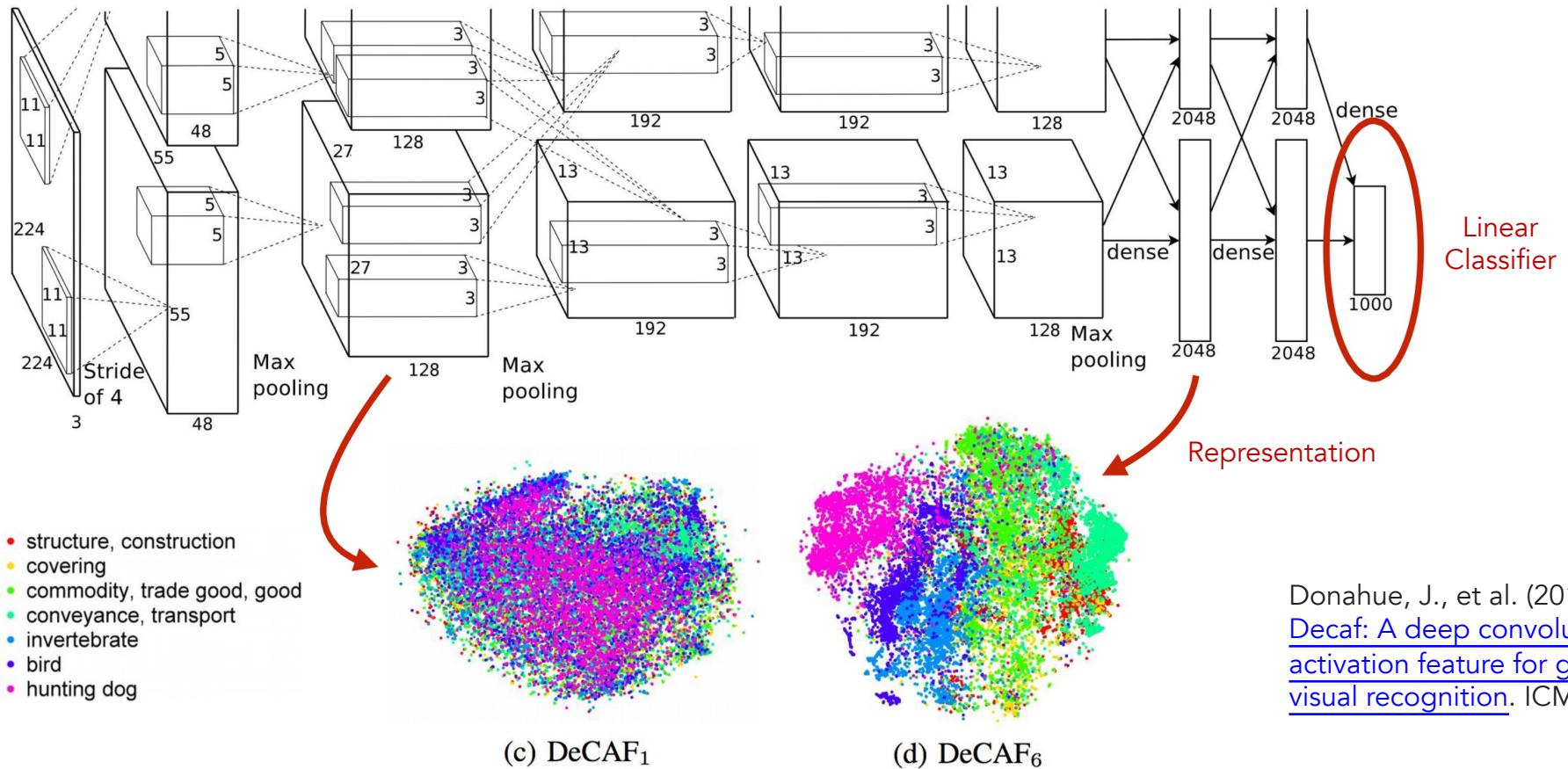


Modality B

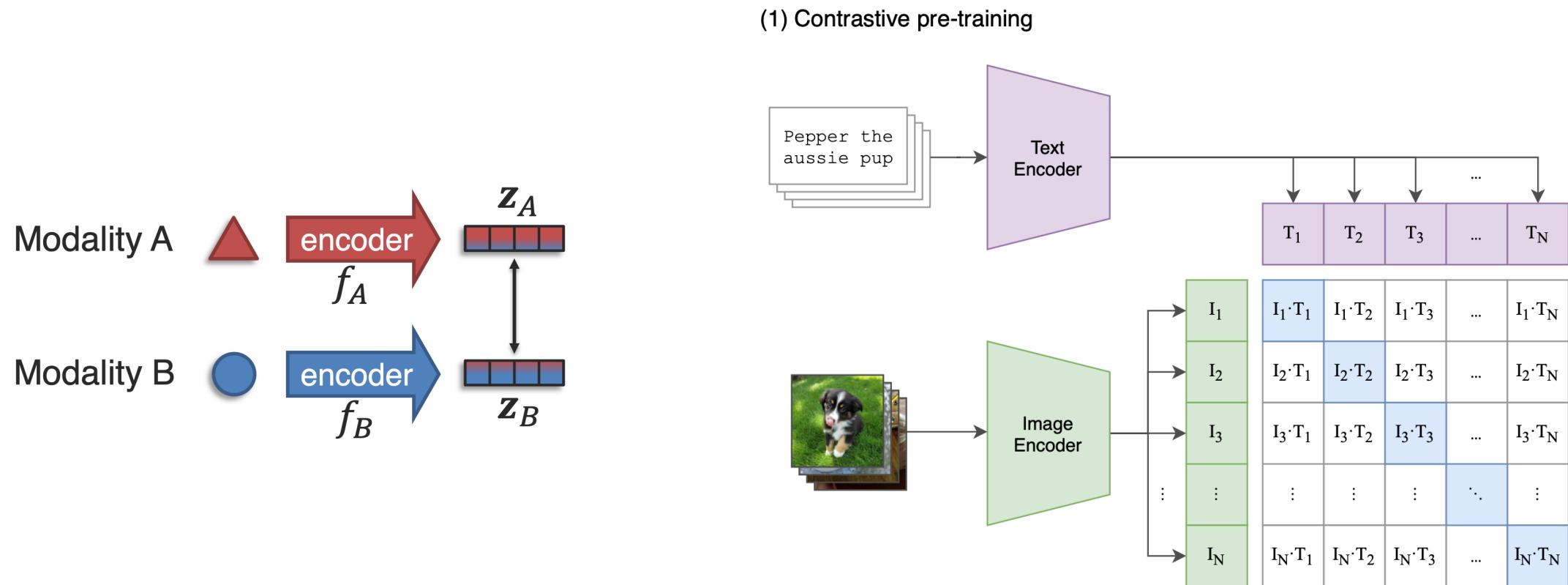


Representation

Instead of completing the task directly, we can also think about how to **learn a good representation** (i.e., embedding) so that a linear classifier can separate the data easily.

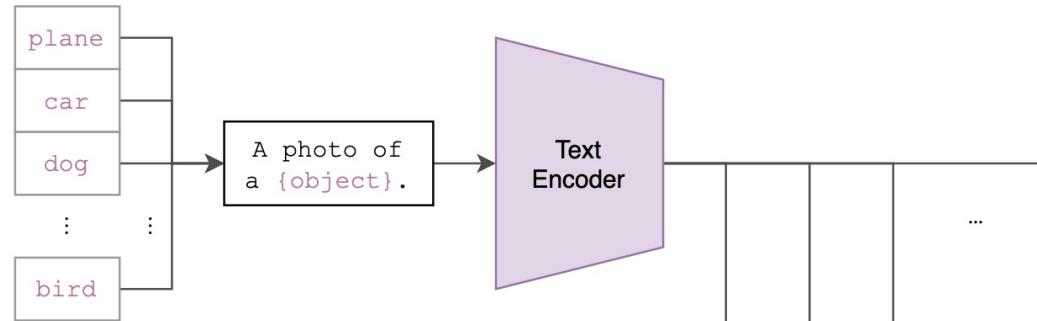


The CLIP model learns a joint text-image representation using a large number of image and text pairs (**vision+language→representation**).

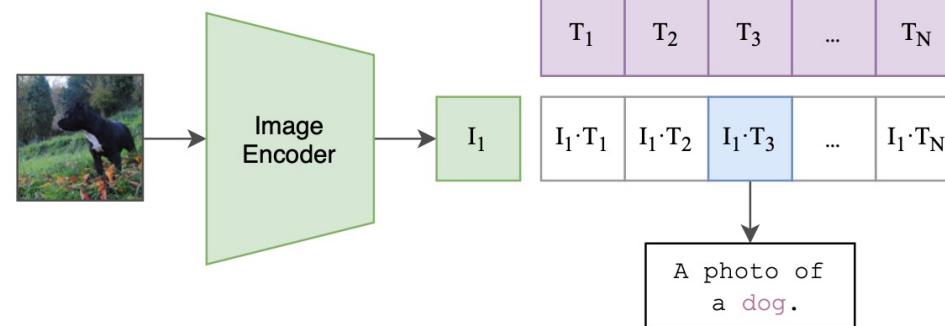


We can use the learned CLIP embedding to perform **zero-shot prediction** by taking the label with the largest similarity score between the label text and the image.

(2) Create dataset classifier from label text

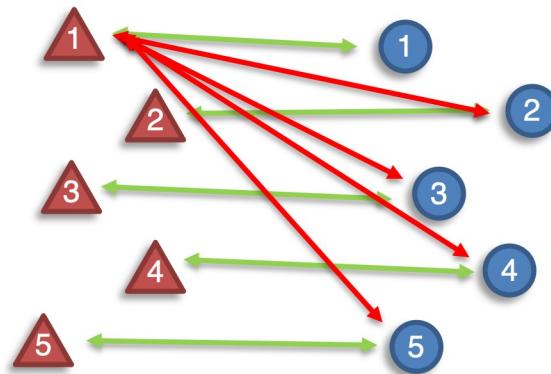


(3) Use for zero-shot prediction



Contrastive Learning brings positive pairs closer and pushes negative pairs far apart.

Paired data: $\{ \triangle, \circ \}$
(e.g., images and text descriptions)



↔ Positive pairs
↔ Negative pairs

Simple contrastive loss:

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

positive pairs

negative pair

Similarity functions are often cosine similarity

Popular contrastive loss: InfoNCE

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^N \log \frac{sim(\mathbf{z}_A^l, \mathbf{z}_B^l)}{\sum_{j=1}^N sim(\mathbf{z}_A^i, \mathbf{z}_B^j)}$$

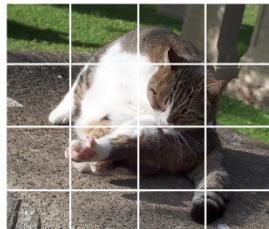
Similarity function can be cosine similarity

positive pairs

negative pairs and positive pairs

One active research area is **foundation models**, which work for both unimodal (e.g., image/text classification) and multimodal tasks (e.g., visual question answering).

MIM, MLM, and MMM Loss:
Hide some data (i.e., masked)
and predict the hidden parts
(image, text, or both)



GC Loss: Contrastive Learning



L_{GC}

ITM Loss: Create a set of
image-text pairs and predict
if an input image matches an
input text



L_{MLM}

This cat was wonderful! He
was making his daily cleaning
on an ancient grave as to say
“I am the boss here!”

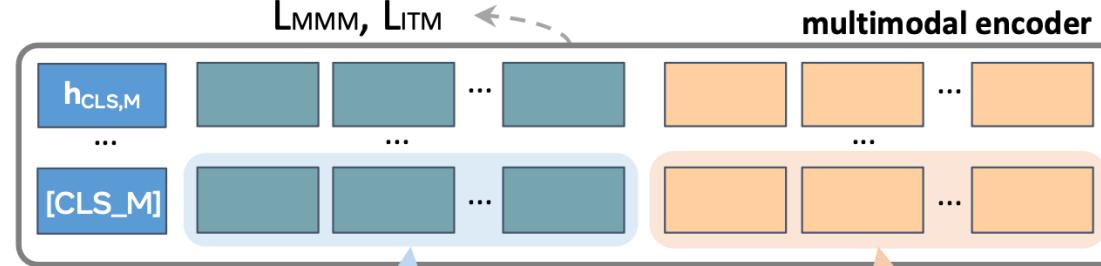
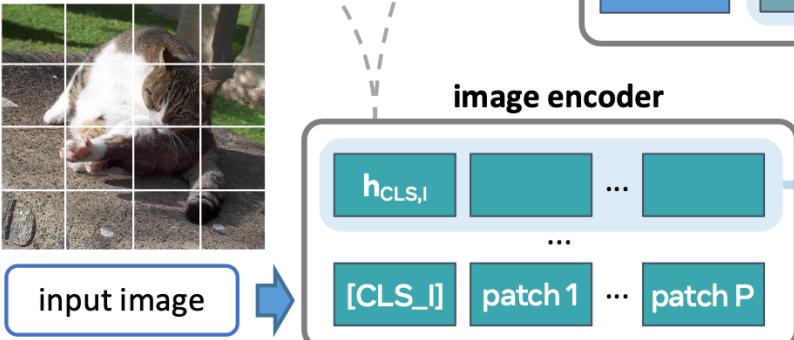


image encoder



L_{MIM}

Take-Away Messages

- Multimodal means having multiple modalities that represent multiple natural phenomena.
- Multiple modalities can exist in different parts of the machine learning pipeline.
- We can fuse the modalities or explicitly learn their connections in the model architecture.
- Self-attention is a way of encoding sequences to tells how much attention each input should pay attention to the other inputs (including itself).
- We can also think about how to learn a good representation (i.e., embedding) so that a linear classifier can separate the data easily.
- Contrastive Learning brings positive pairs closer and pushes negative pairs far apart.



Questions?