Data Science

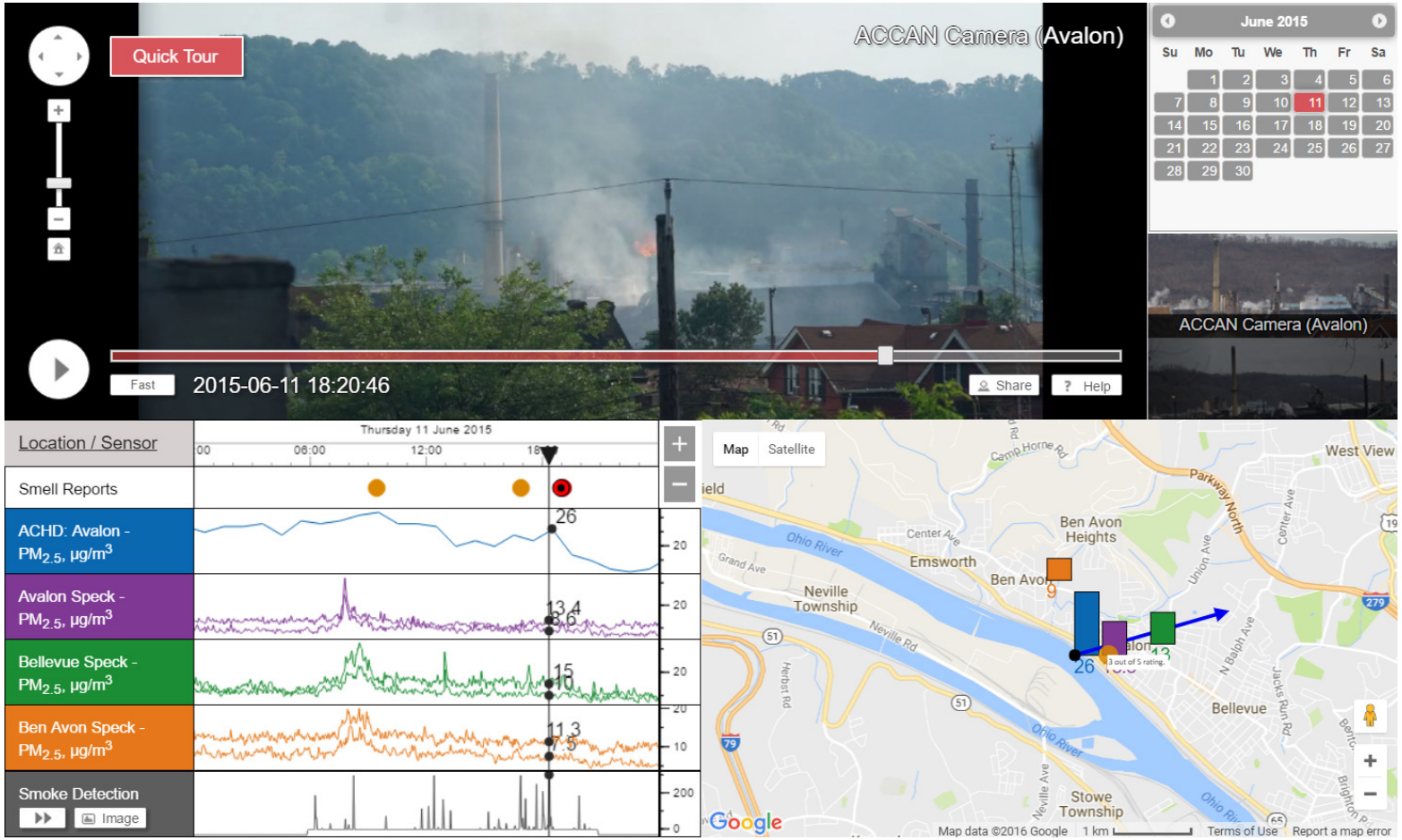
Lecture 11: Multimodal Data Processing

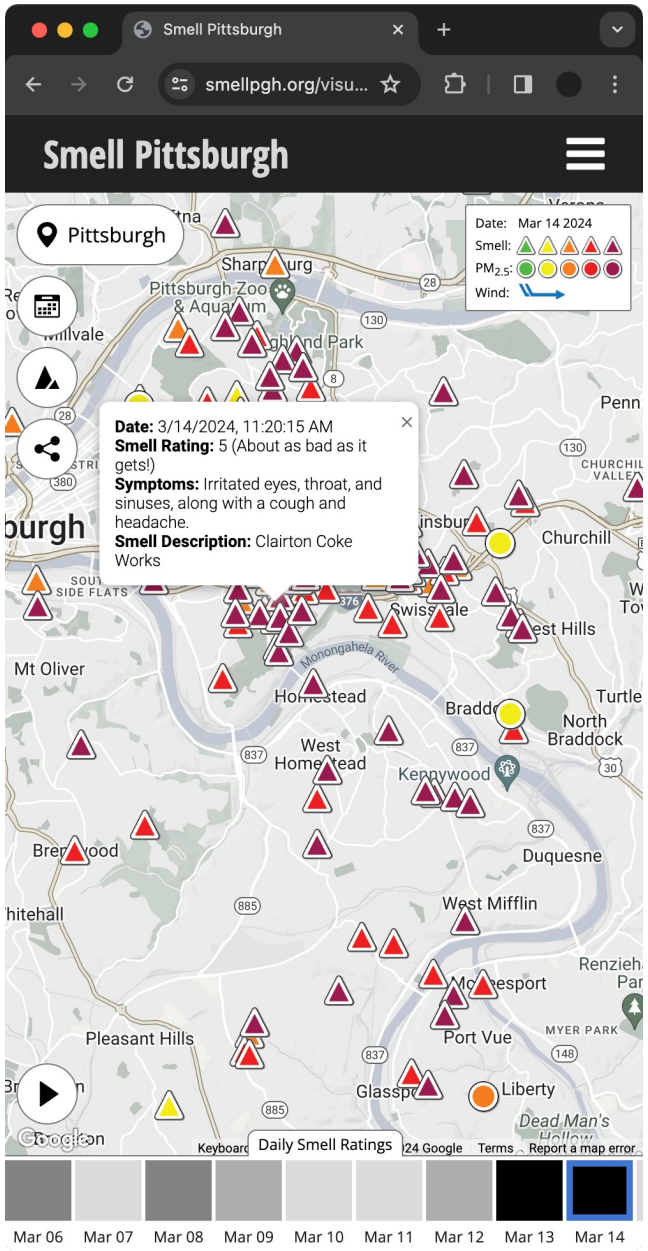
University of Amsterdam

Lecturer: Yen-Chia Hsu

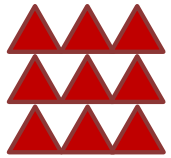
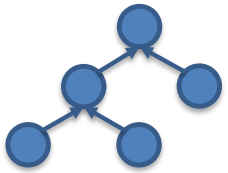
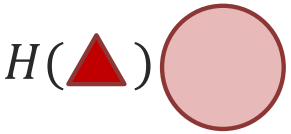
Date: Oct 2025

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



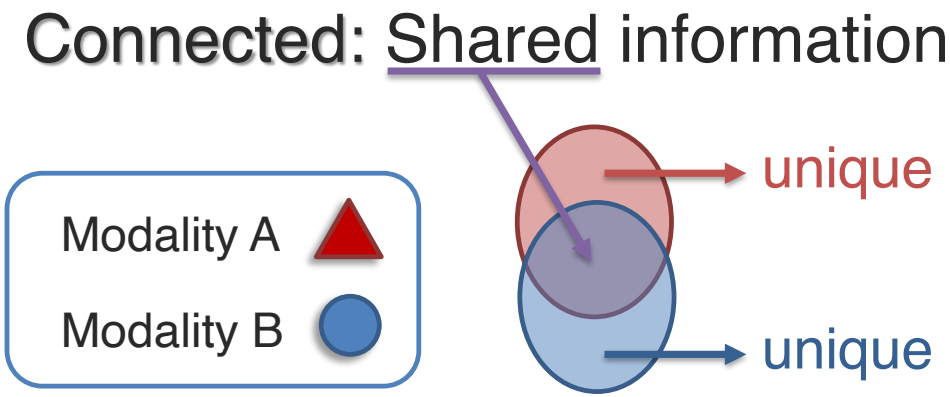


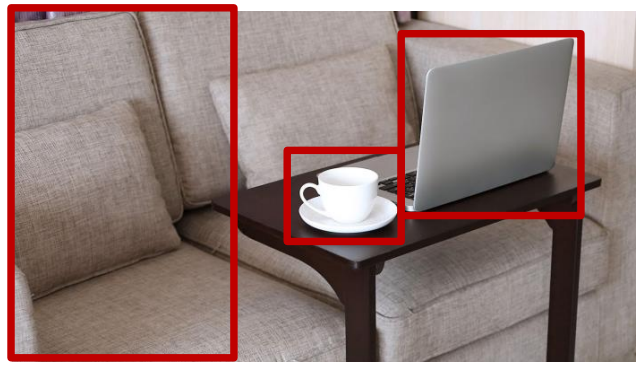
Different modalities can have different characteristics.

	Modality A (e.g., vision)	Modality B (e.g., language)	Examples (e.g., vision/language)
1 **Element representations:** Discrete, continuous, granularity			• Sets of images • Sets of characters
2 **Element distributions:** Density, frequency			• Pixel density • Word frequency
3 **Structure:** Temporal, spatial, latent, explicit			• Nearby objects • Semantics
4 **Information:** Abstraction, entropy			• Ordering of words • Motion, color
5 **Noise:** Uncertainty, noise, missing data			• Occlusion • Ambiguity
6 **Relevance:** Task, context dependence			• Object detection • Machine translation

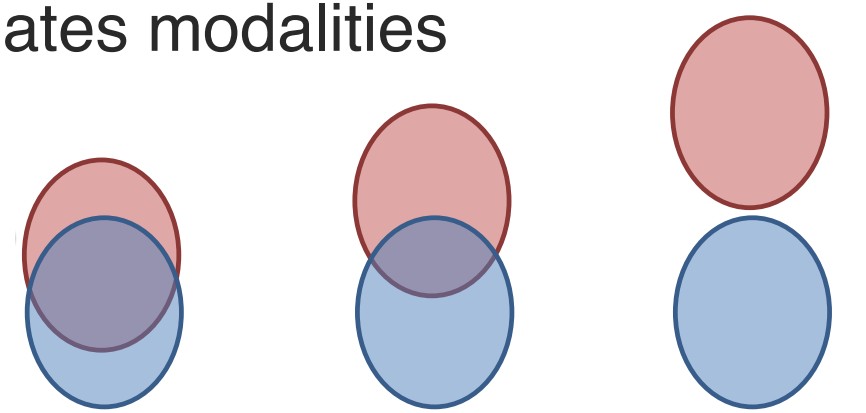
Source: CMU 11-777 MMML Course Lecture 1.1 -- https://cmu-multicomp-lab.github.io/mmml-course/fall2023/schedule/

Different modalities can share information with different levels of connections.





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



strong connection

weak connection

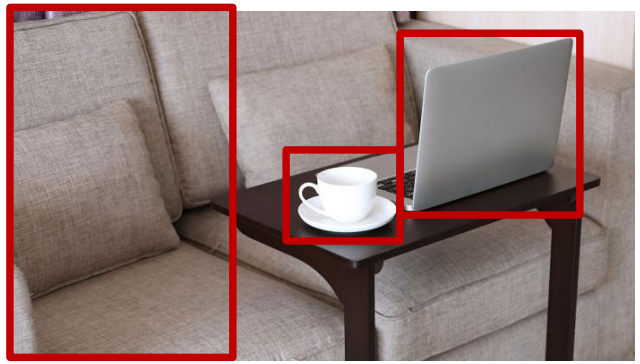
no connection

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

The shared information can be connected in different ways.

Association

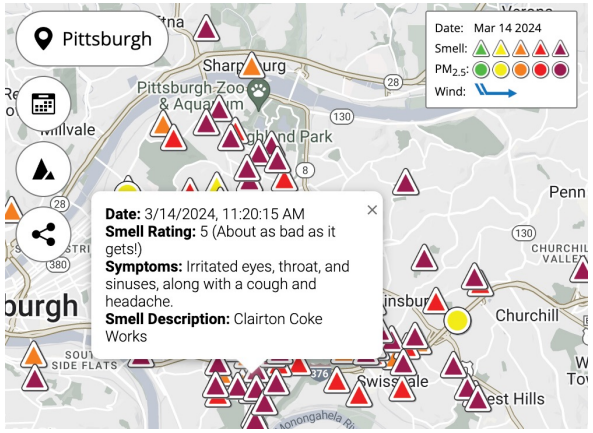
e.g., correlation, co-occurrence



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

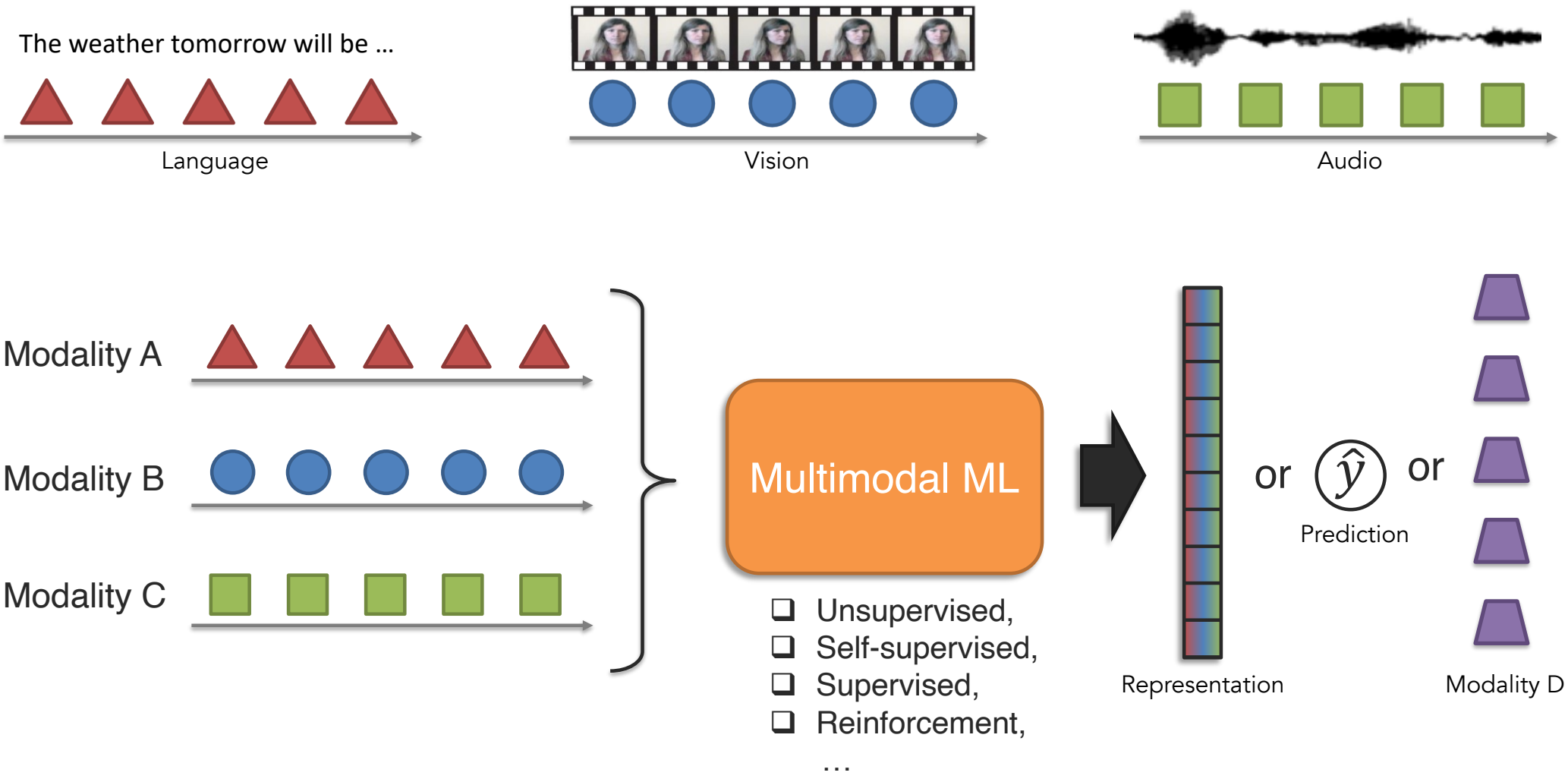
Dependency

e.g., causal, temporal



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

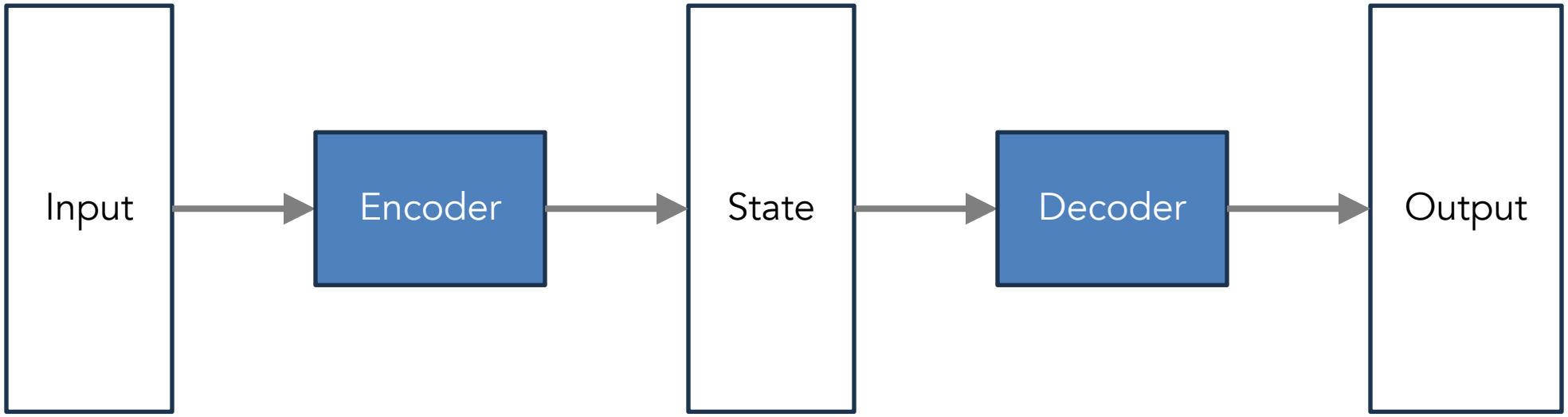
Multiple modalities can exist in different parts of the machine learning pipeline.



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

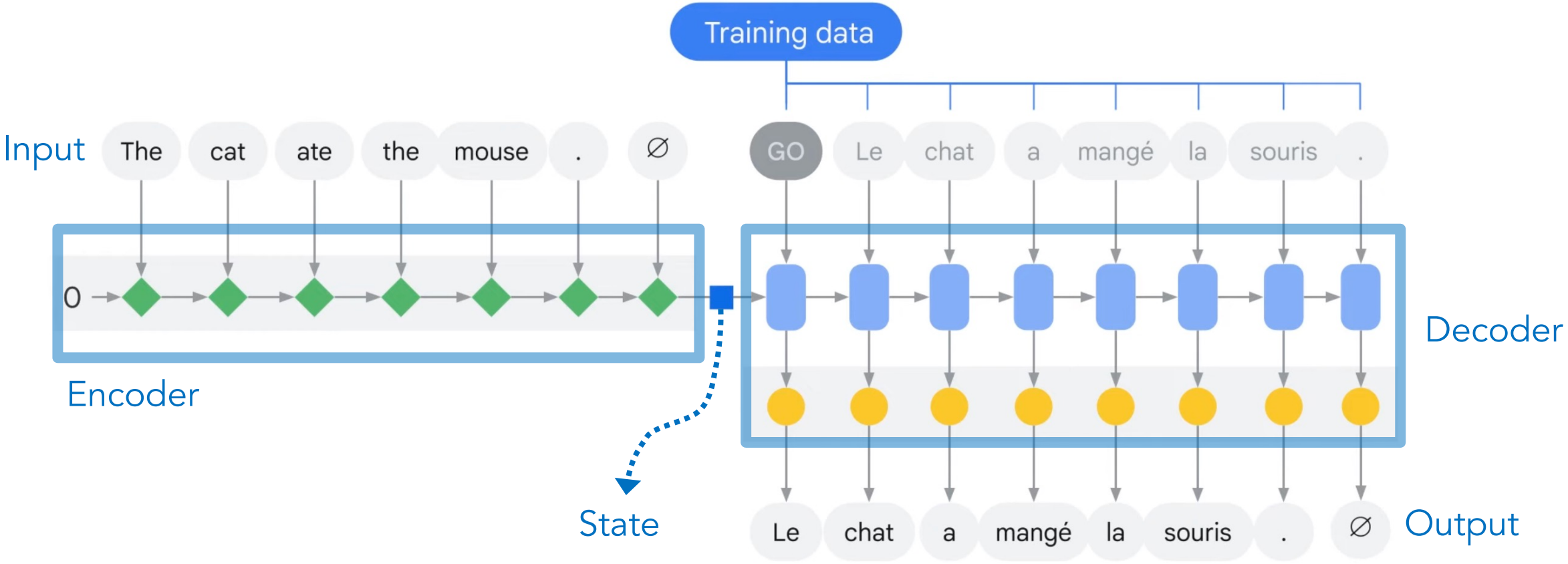

Modality A
Modality B

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.



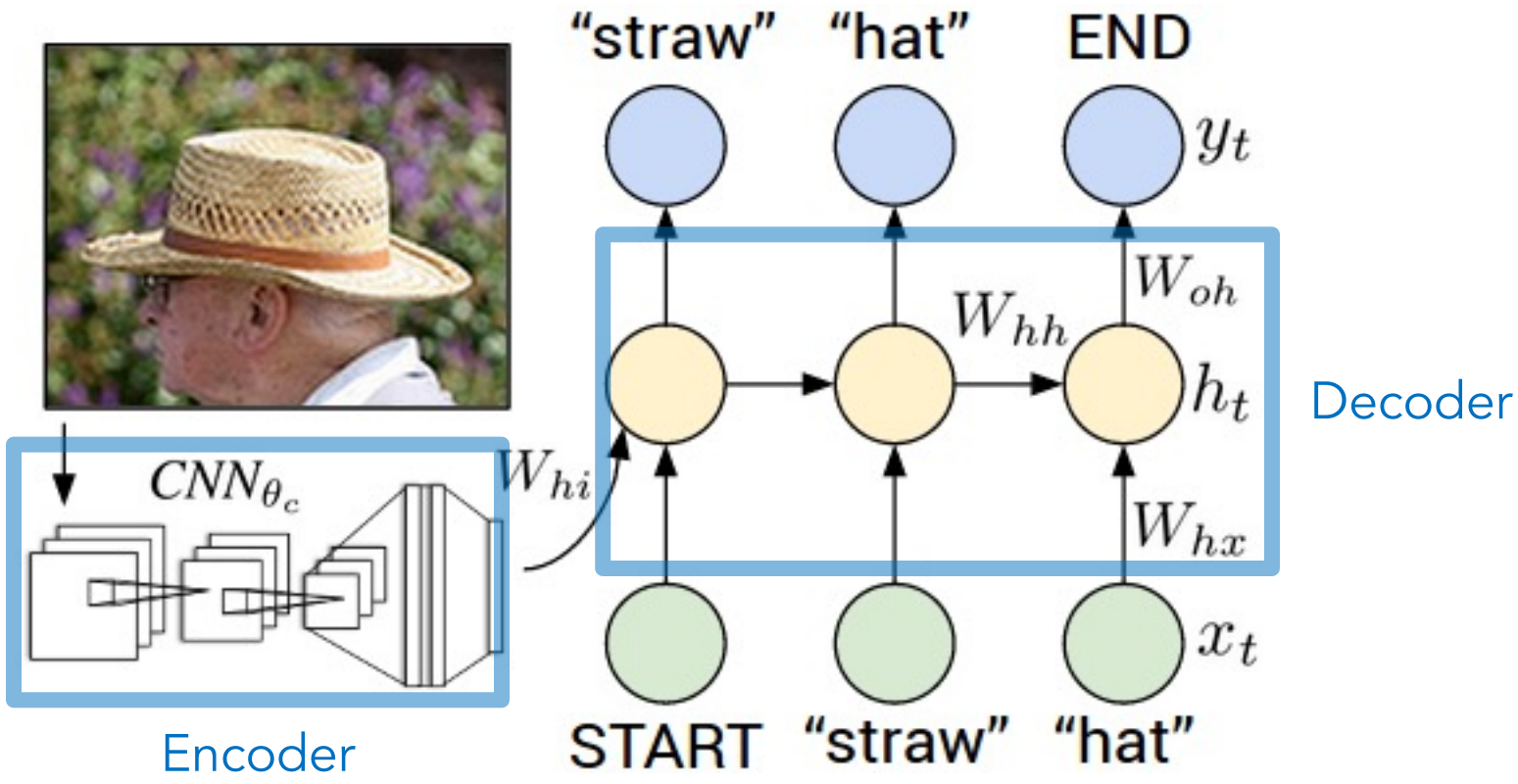
More about the encoder-decoder architecture -- https://d2l.ai/chapter_recurrent-modern/encoder-decoder.html

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.



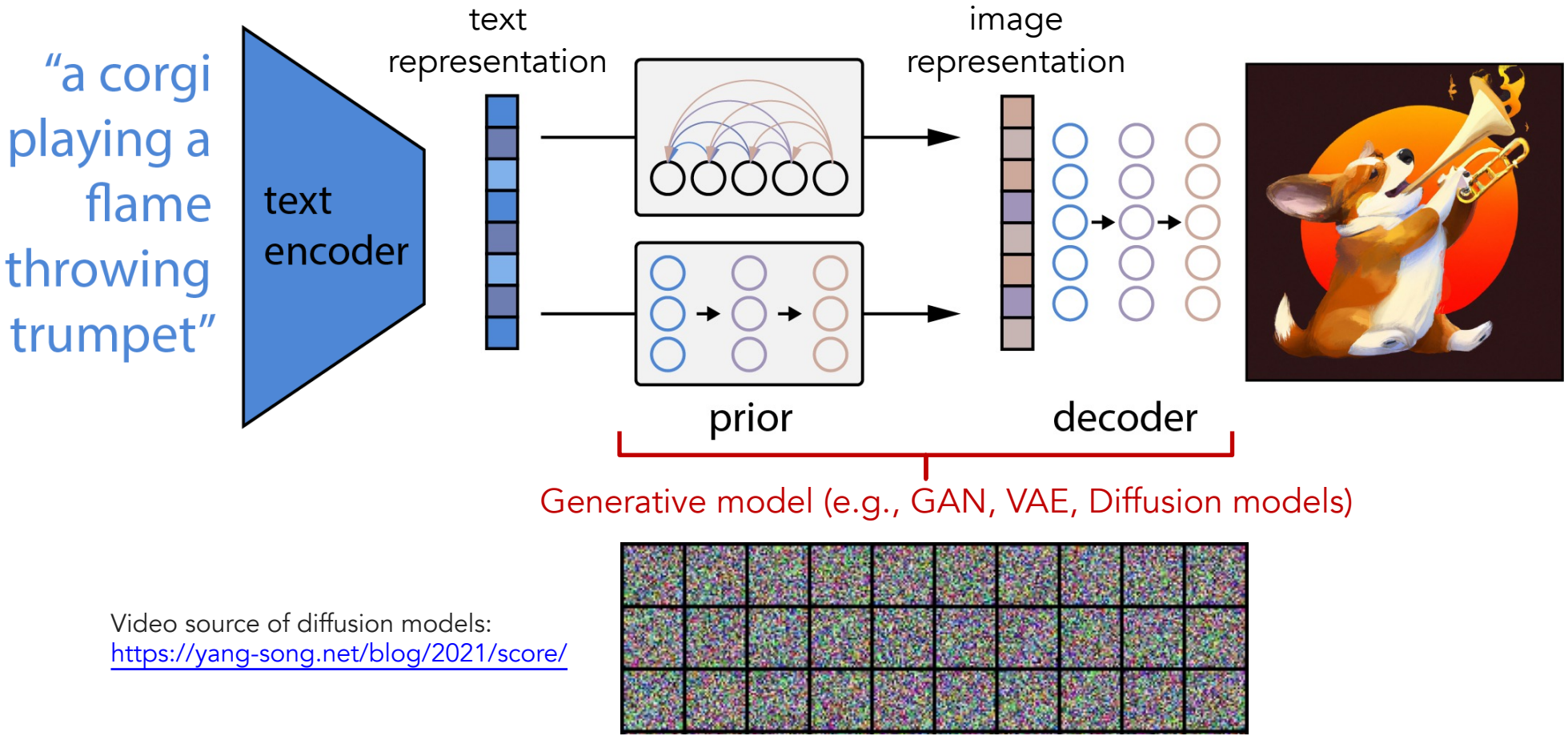
Explanation of encoder-decoder architecture in a single modality setting -- https://www.youtube.com/watch?v=zbdong_h-x4

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).



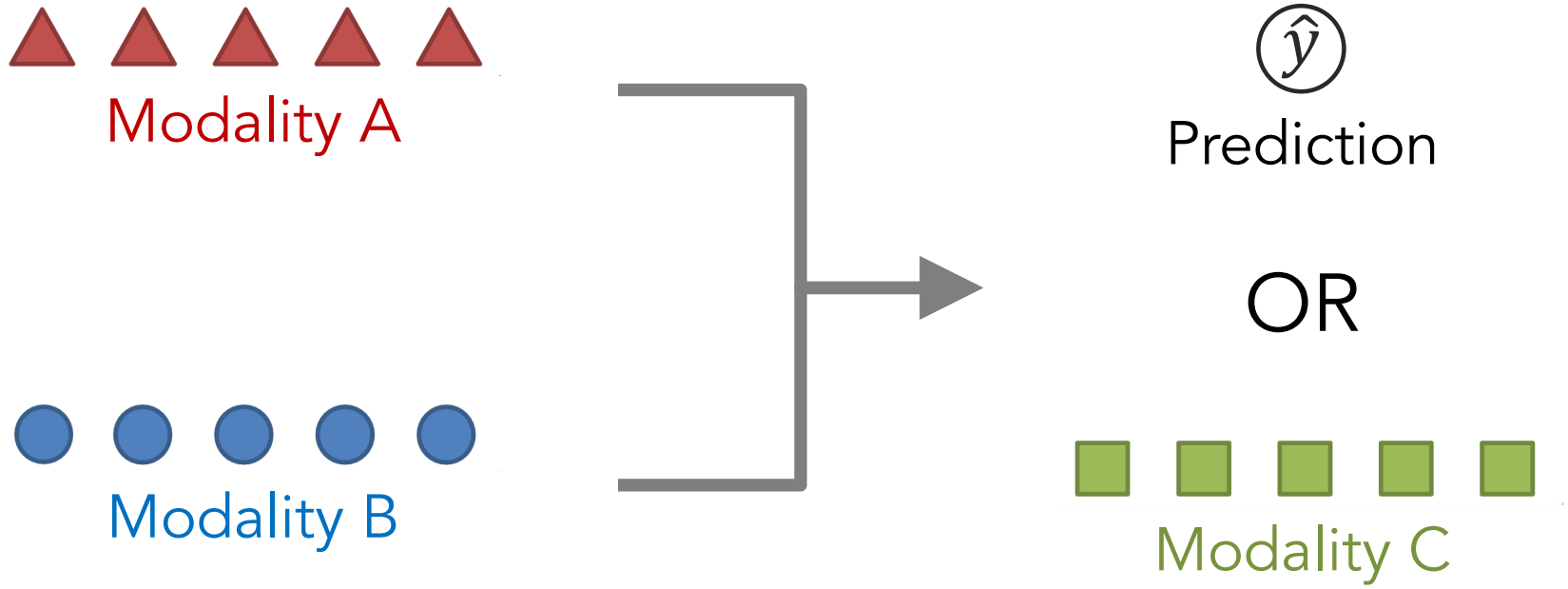
Karpathy, A., & Fei-Fei, L. (2015). Deep Visual-Semantic Alignments for Generating Image Descriptions. CVPR.

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

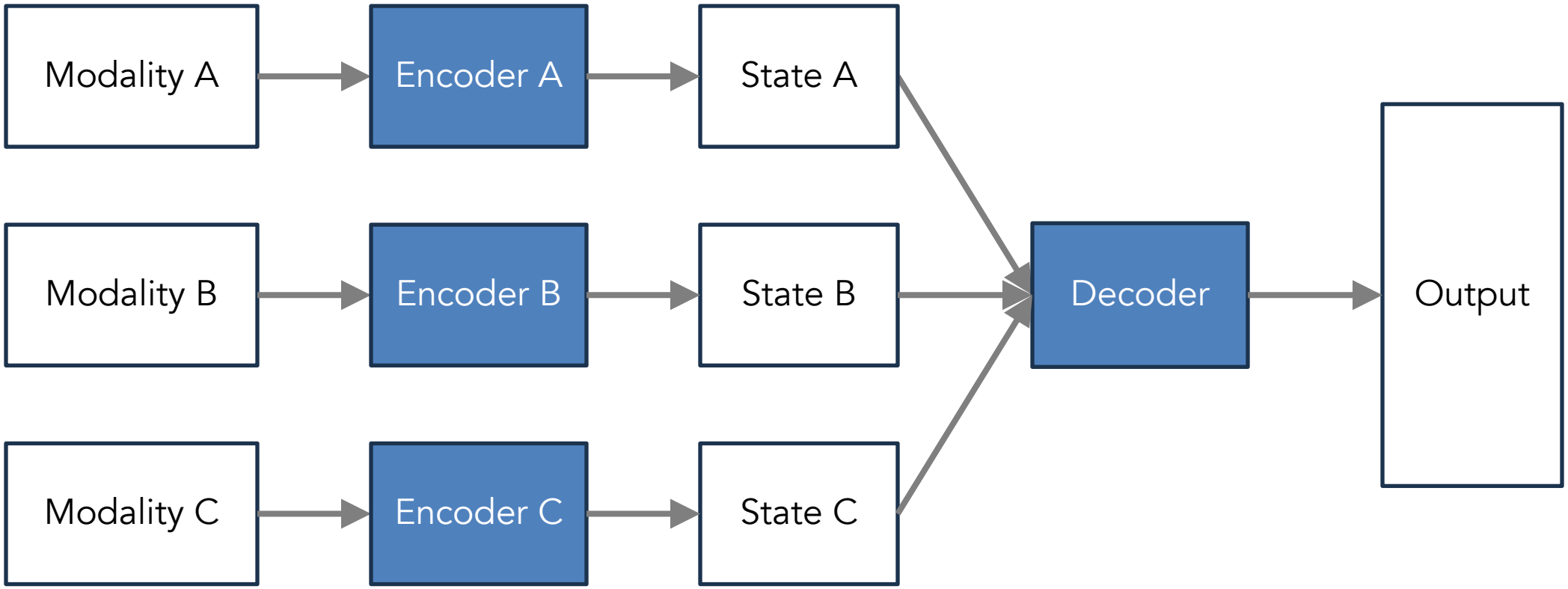


Video source of diffusion models:
https://yang-song.net/blog/2021/score/

Ramesh, A., et al. (2022). Hierarchical Text-Conditional Image Generation with CLIP Latents. arXiv.

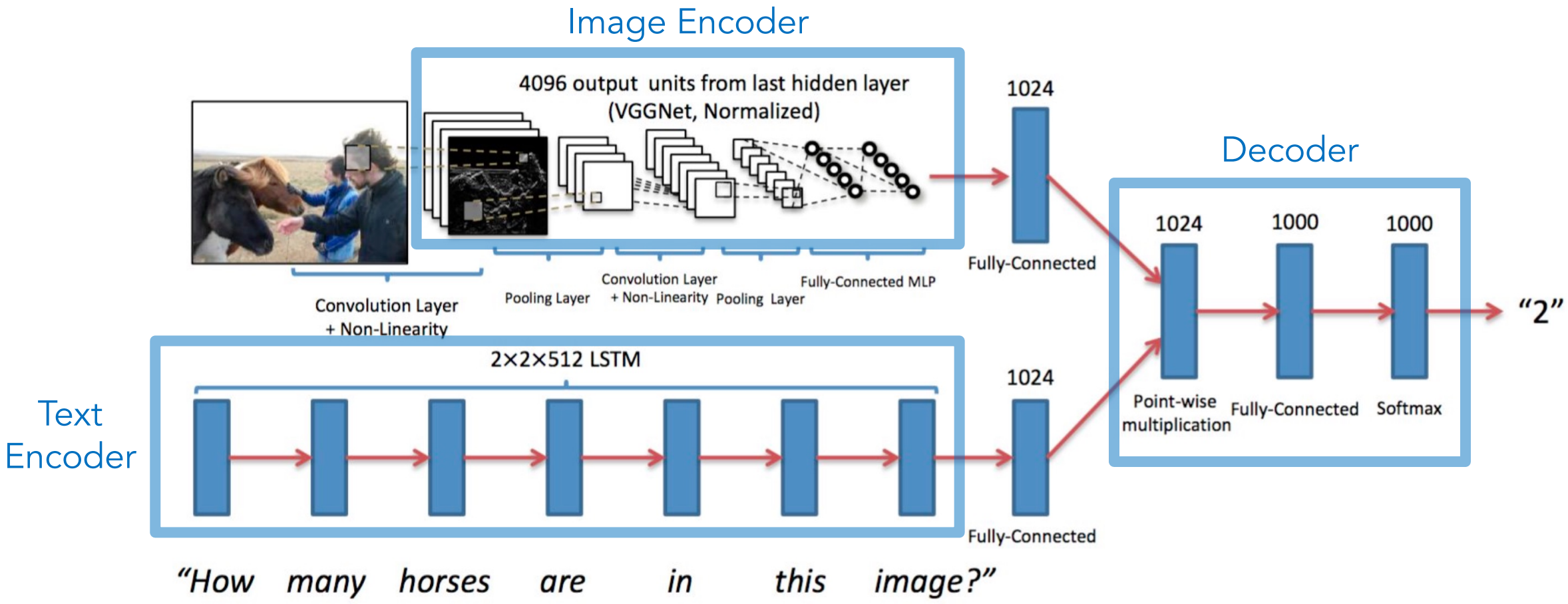

Modality A
Modality B
$\hat{y}$
Prediction
OR
Modality C

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.



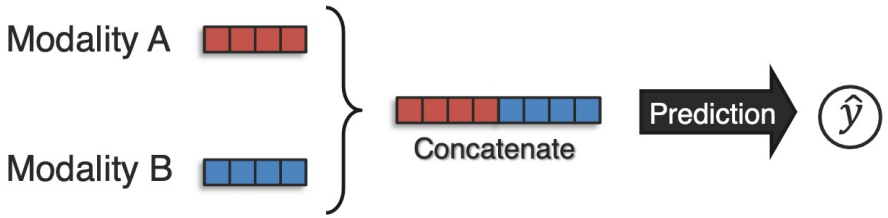
More about the encoder-decoder architecture -- https://d2l.ai/chapter_recurrent-modern/encoder-decoder.html

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

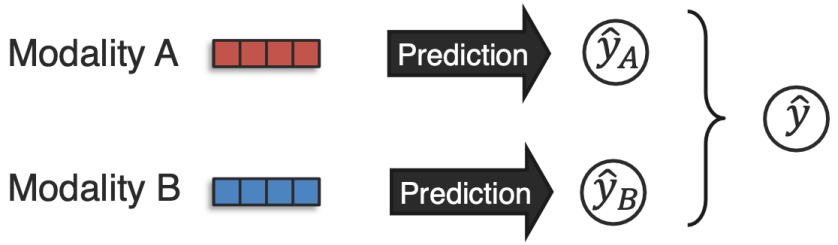


Zhu, Y., et al. (2016). Visual7W: Grounded Question Answering in Images. CVPR. (Figure is from Stanford cs231n course)

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

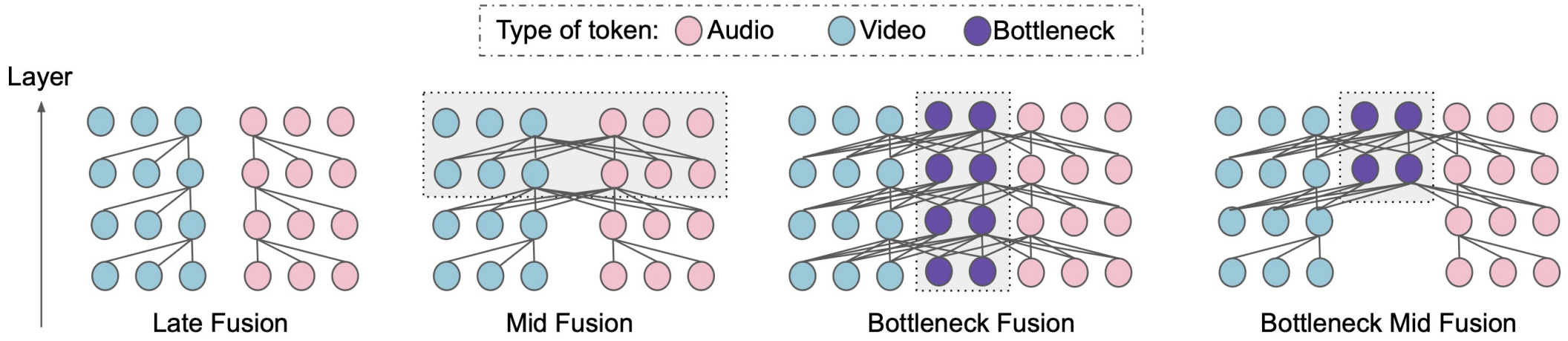


Early Fusion



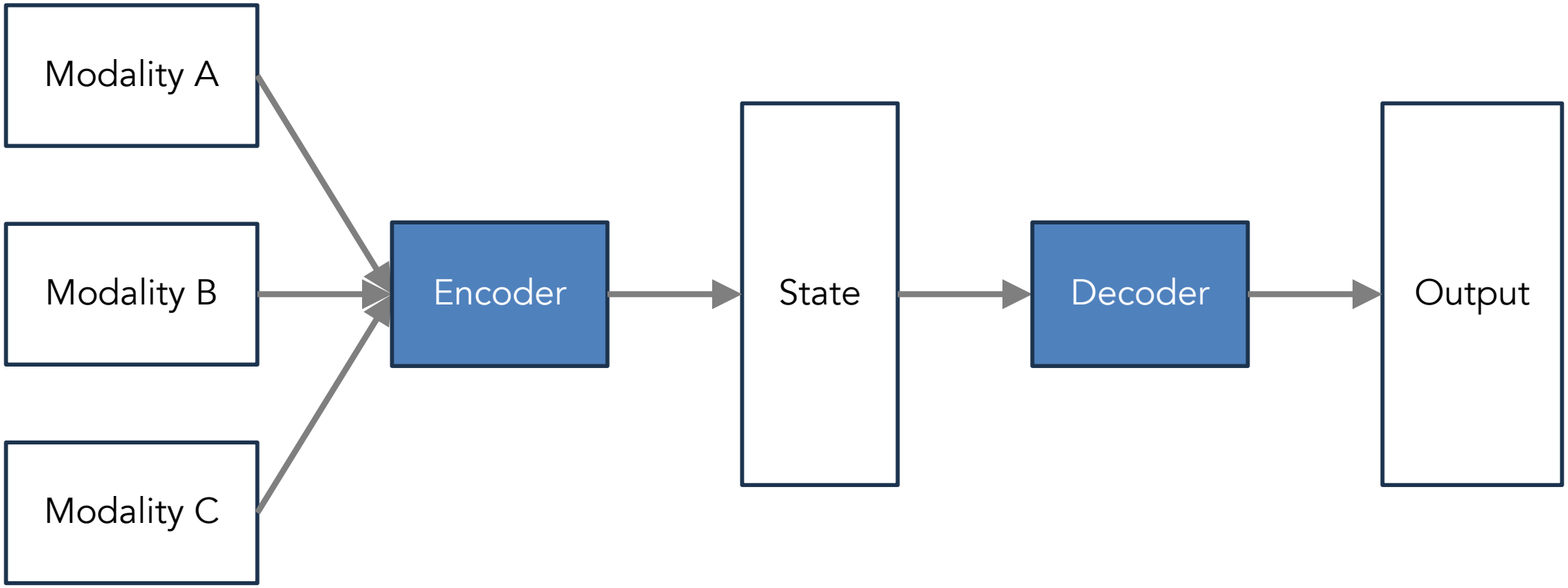
Late Fusion

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



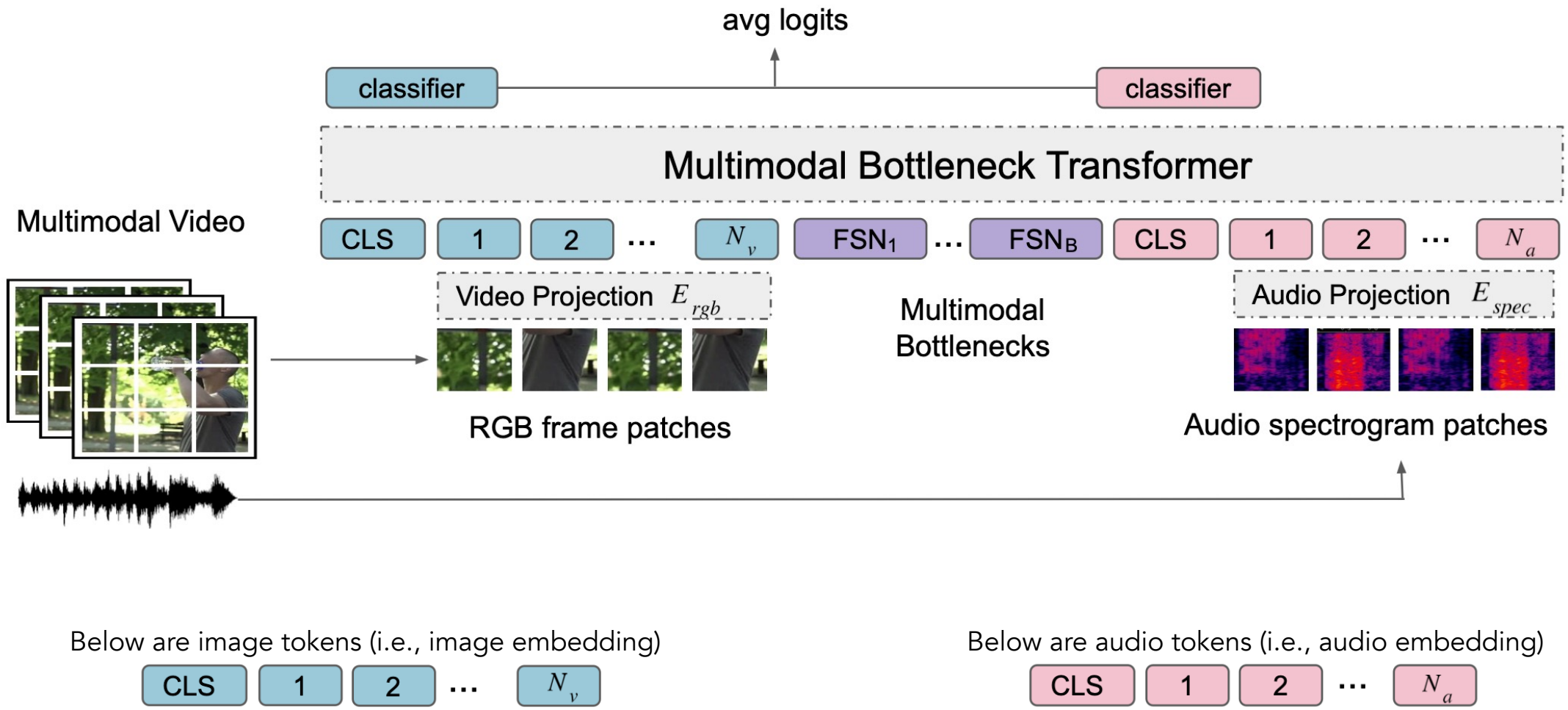
Nagrani, A., et al. (2021). Attention Bottlenecks for Multimodal Fusion. NeurIPS

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.



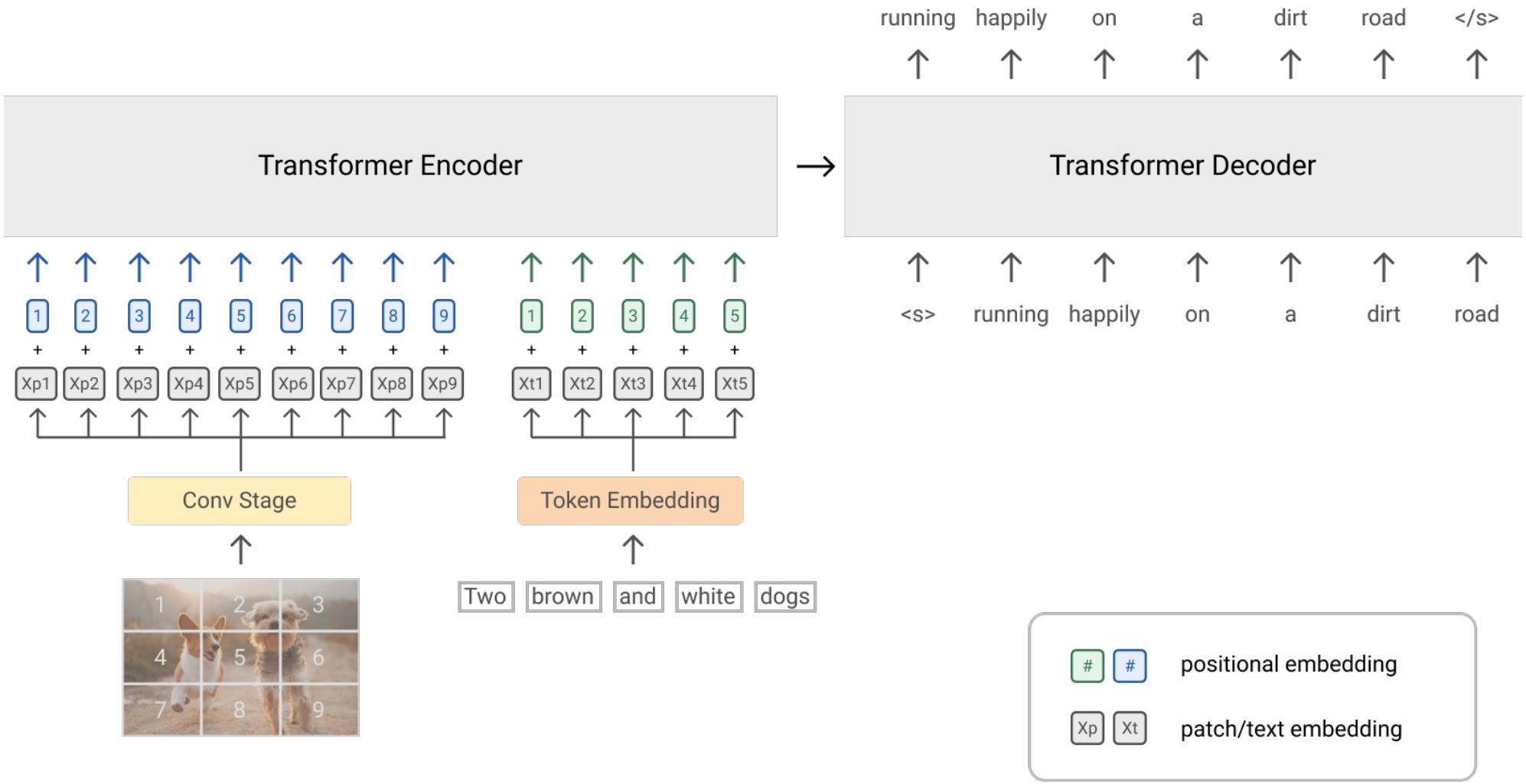
More about the encoder-decoder architecture -- https://d2l.ai/chapter_recurrent-modern/encoder-decoder.html

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



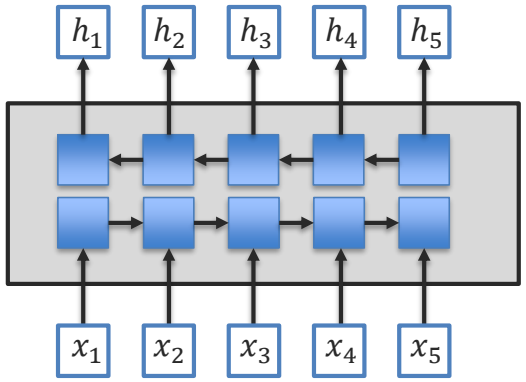
Nagrani, A., et al. (2021). Attention Bottlenecks for Multimodal Fusion. NeurIPS

Transformers also work for multiple vision-language tasks (vision+language→language).

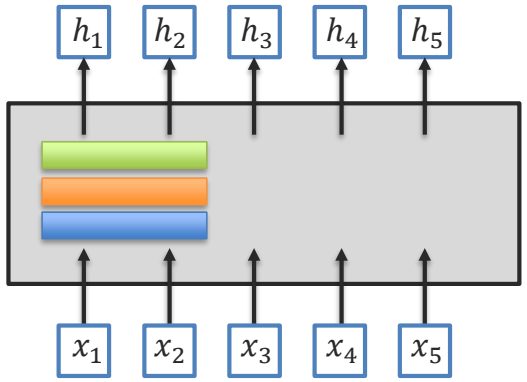


Wang, Z., et al. (2021). SimVLM: Simple Visual Language Model Pretraining with Weak Supervision. ICLR.

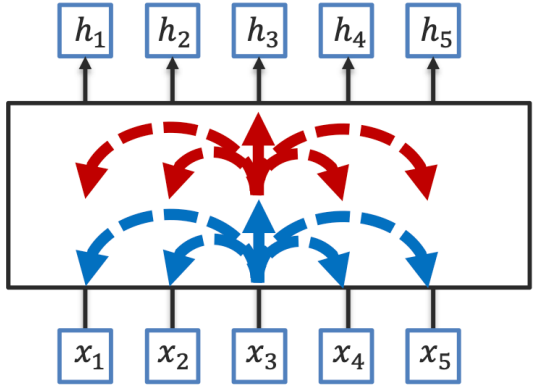
Transformers use self-attention, which is a way of encoding sequences to tells how much attention each input should pay attention to the other inputs (including itself).



Bidirectional RNN



Convolution



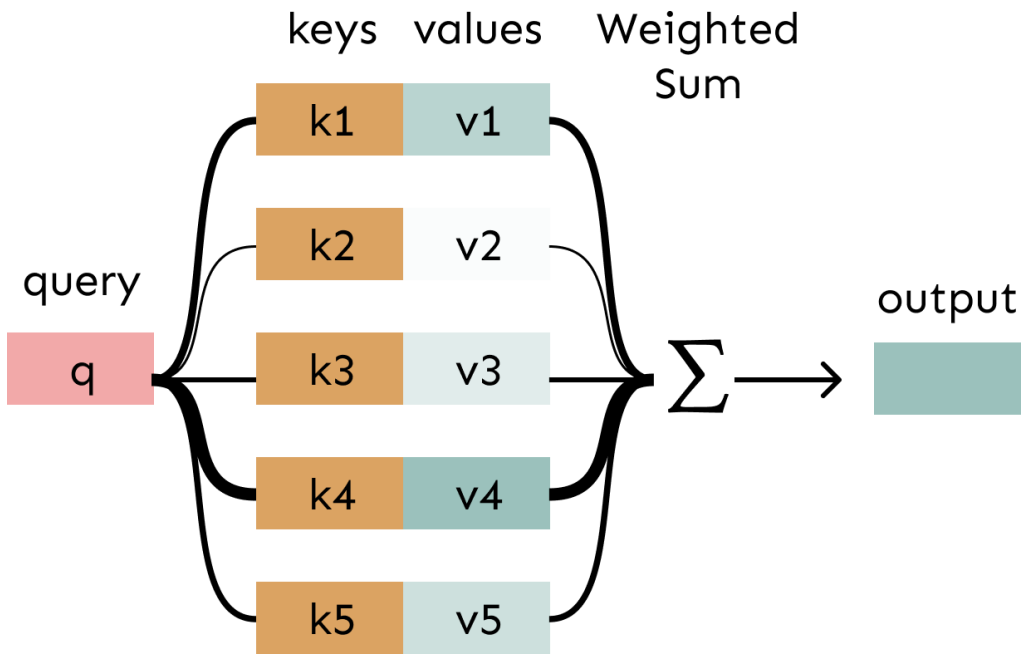
Self-attention

Source: CMU 11-777 MMML Course Lecture 4.2 -- https://cmu-multicomp-lab.github.io/mmml-course/fall2023/schedule/

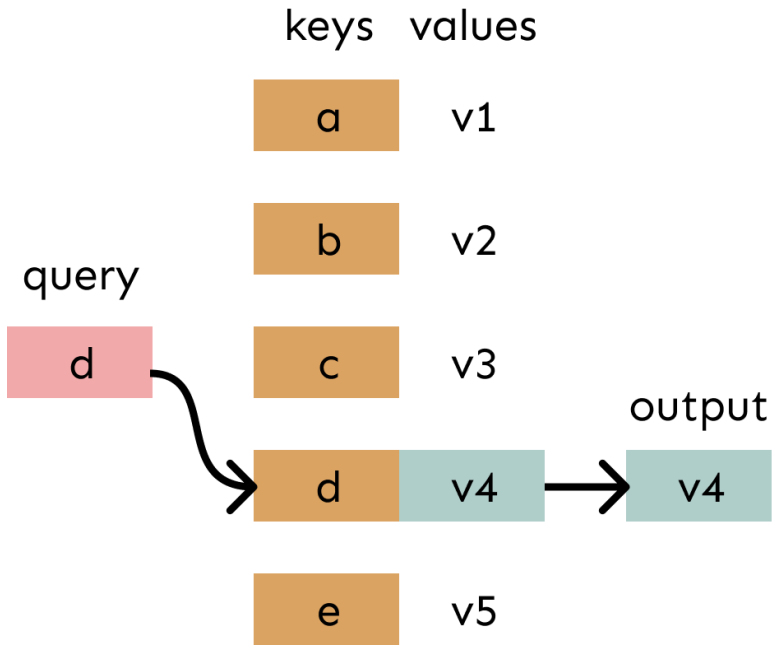
Attention is *weighted* averaging, which lets you do lookups!

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.



The figure is from the Stanford CS224N course -- https://web.stanford.edu/class/cs224n/

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):

$$e_t = score(q, k_t) = q^T k_t$$

W_k is trainable

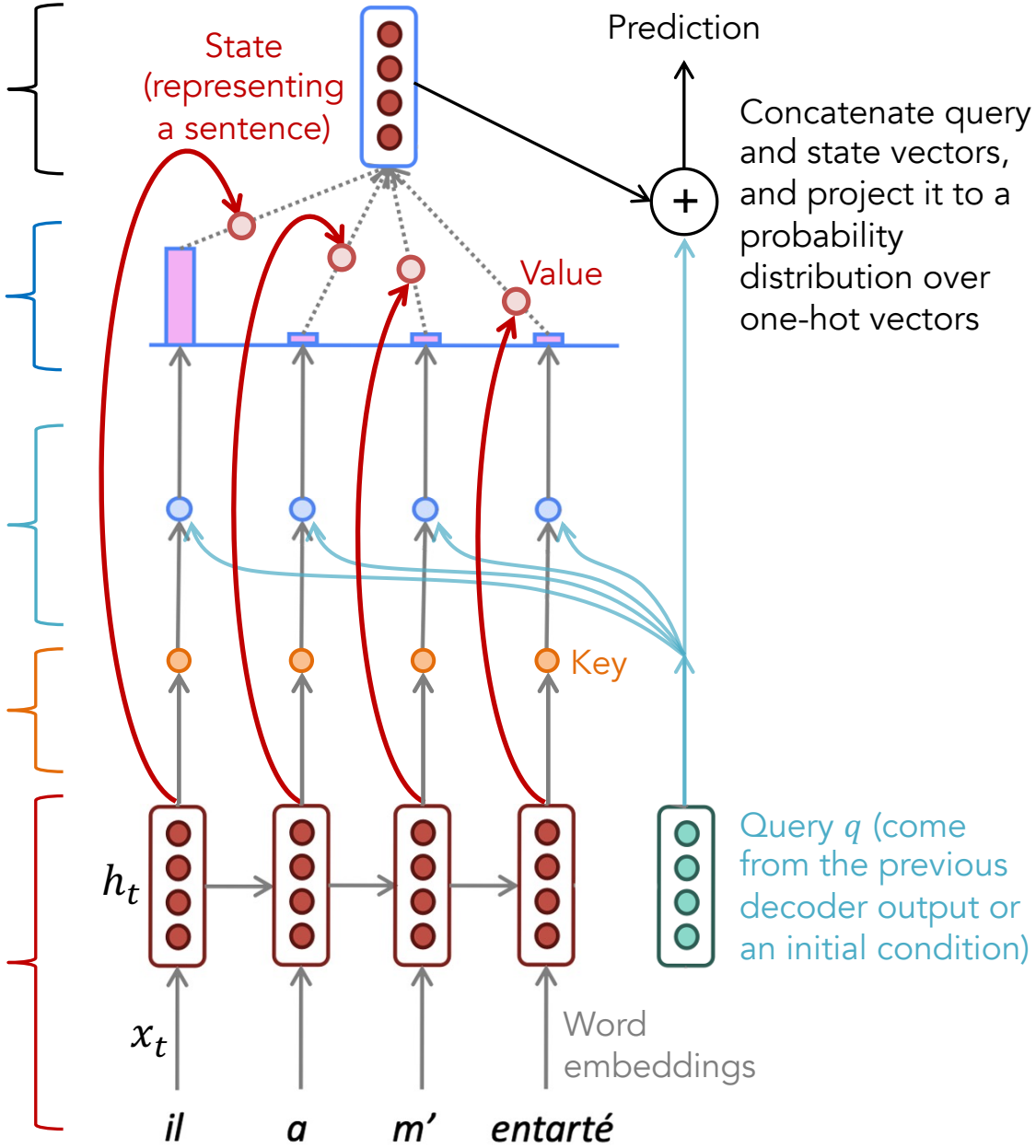
Step 2: Transform encoder outputs (dimension reduction):

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

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



Source -- https://web.stanford.edu/class/cs224n/

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:

$$e_t = score(q, k_t) = q^T k_t$$

W_q , W_k , and W_v are trainable

Step 2: Compute the key vector

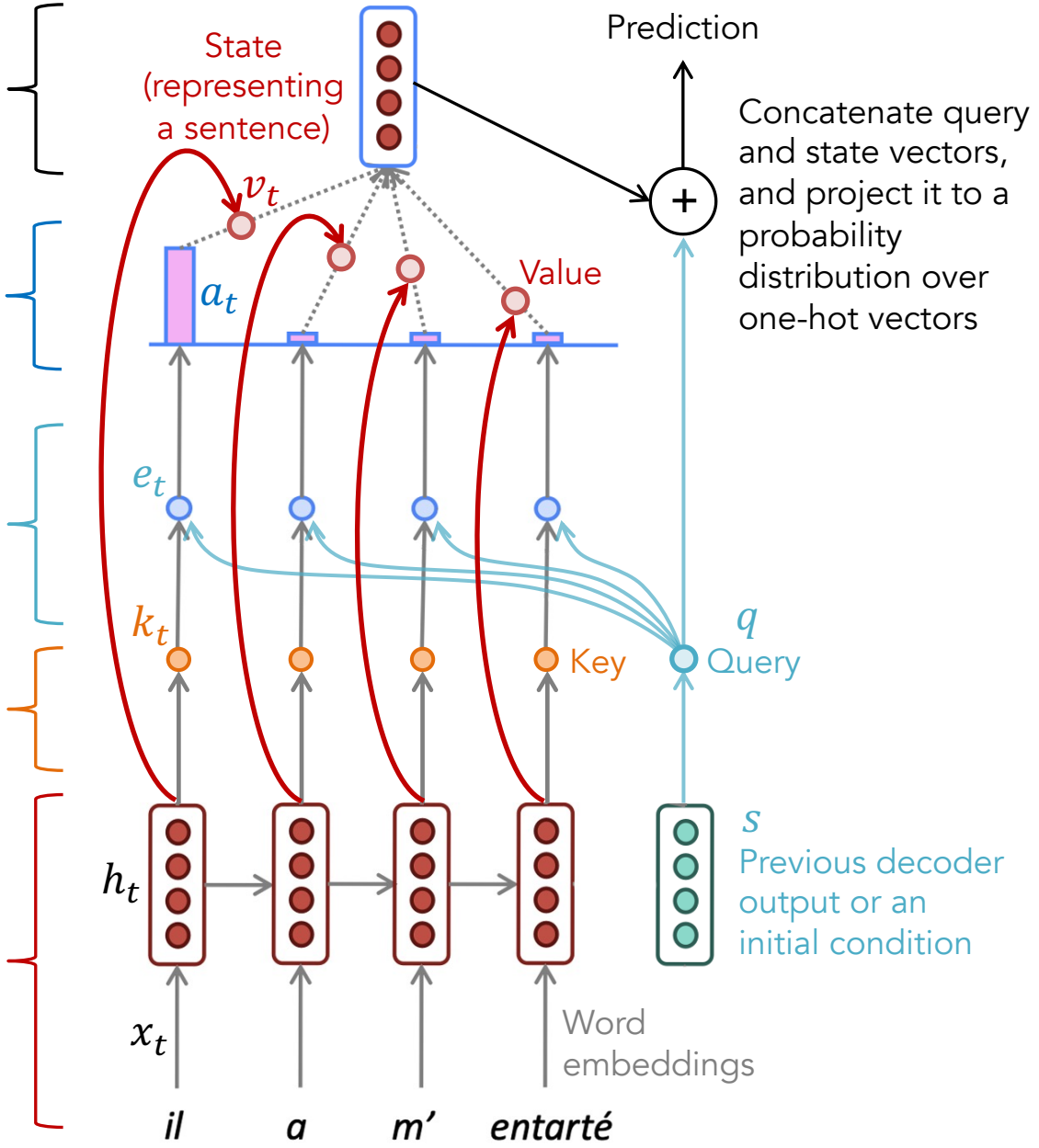
$$k_t = \tanh(W_k h_t)$$

Step 1: Compute the value and query vectors

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

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

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

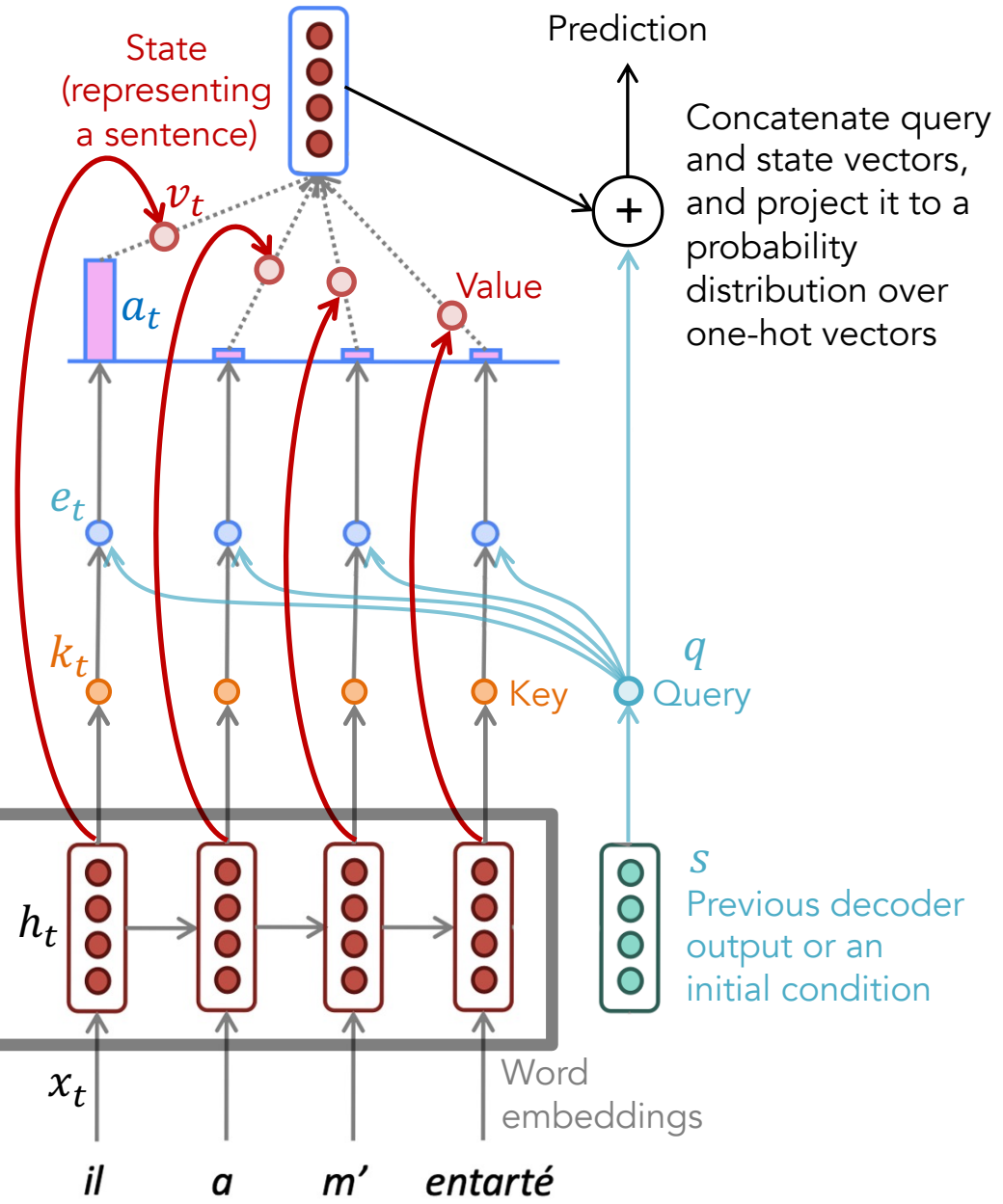


Source -- https://web.stanford.edu/class/cs224n/

Attention addresses the problem of remembering previous information.

But there is still one problem.

Why is the recurrent neural network structure not desirable?



Source -- https://web.stanford.edu/class/cs224n/

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:

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

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

Step 2: Compute the key vector

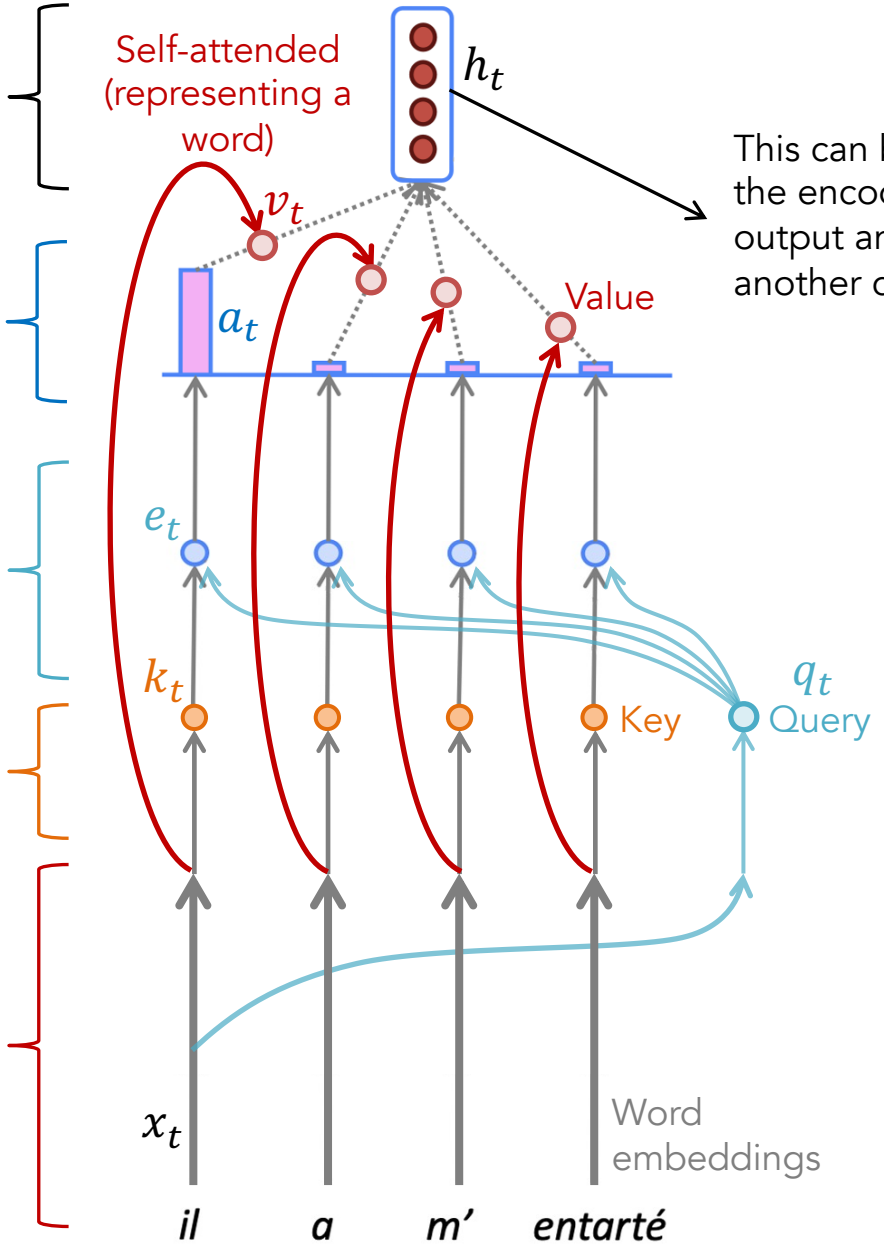
$$k_t = W_k x_t$$ [there is no tanh function now]

Step 1: Compute the value and query vectors

$$v_t = W_v x_t$$

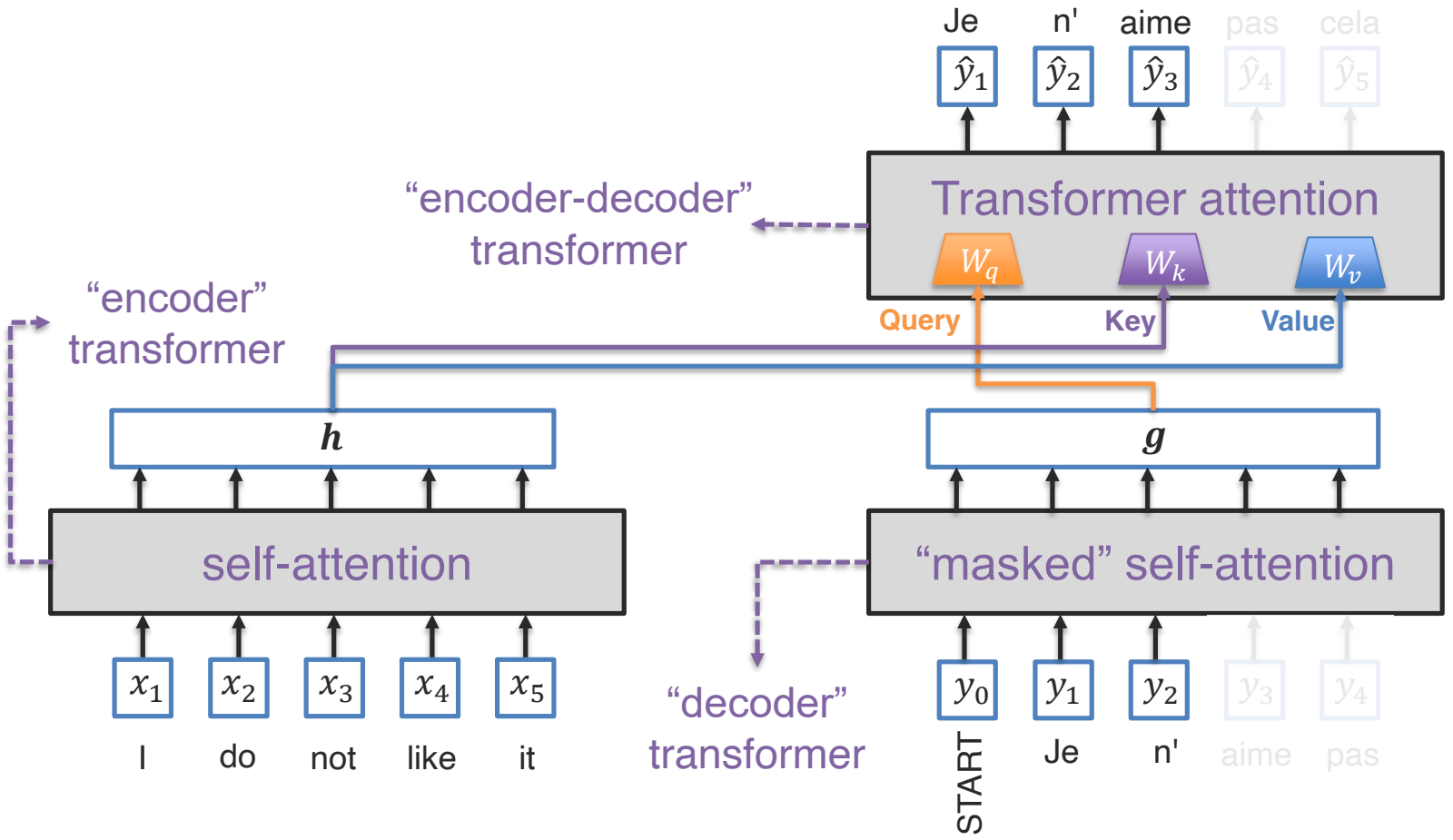
$$q_t = W_q x_t$$

(x_t is the word embedding vector)



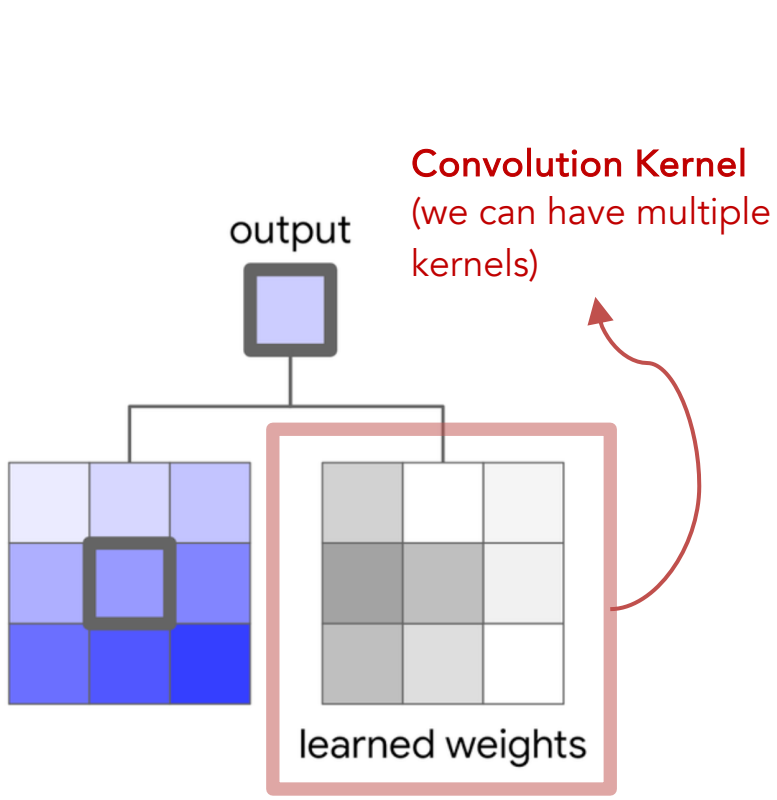
Source -- https://web.stanford.edu/class/cs224n/

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).

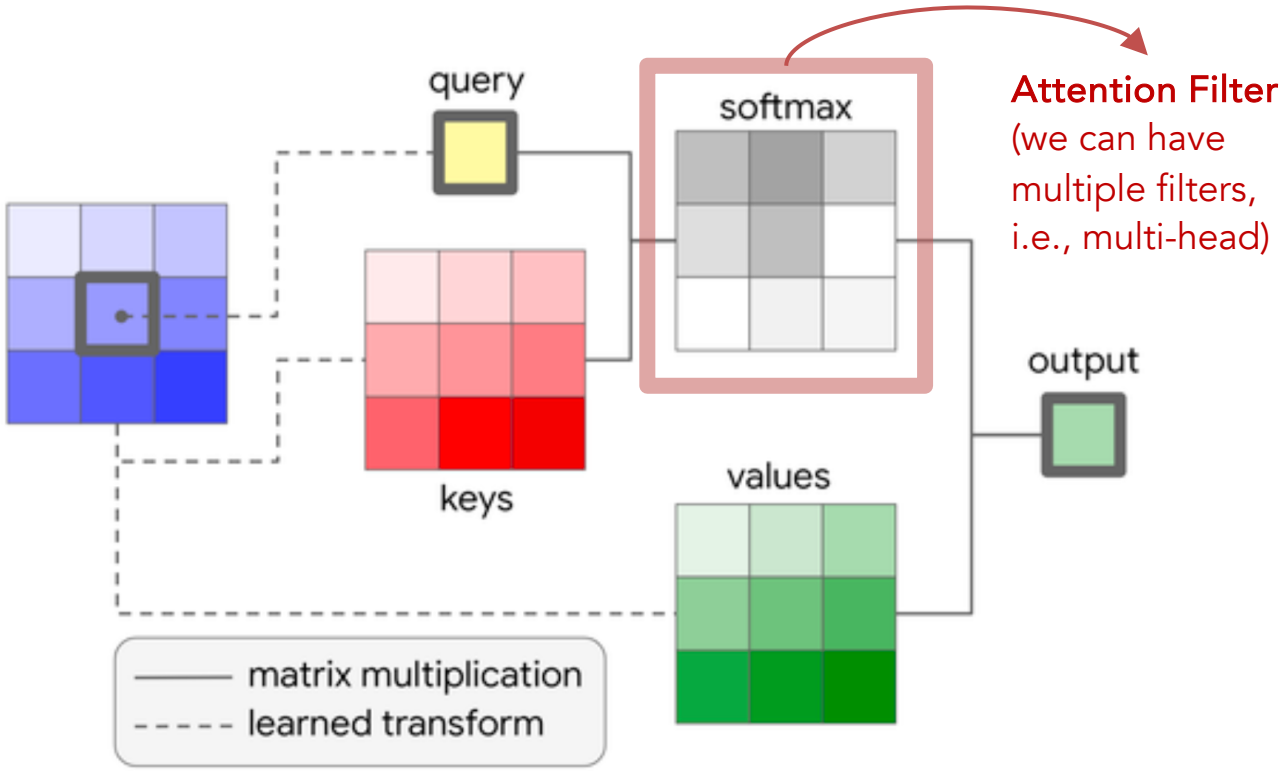


Source: CMU 11-777 MMML Course Lecture 4.2 -- https://cmu-multicomp-lab.github.io/mmml-course/fall2023/schedule/

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.



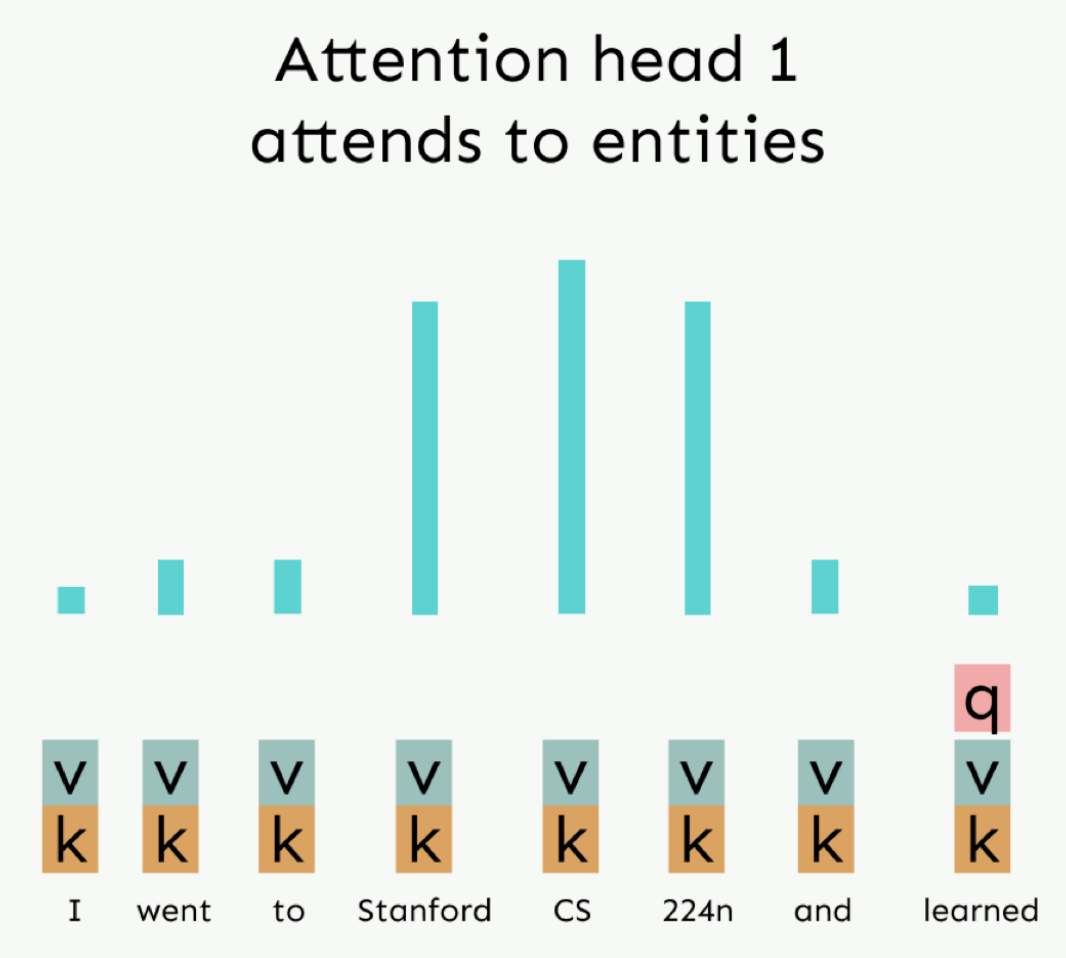
Convolution

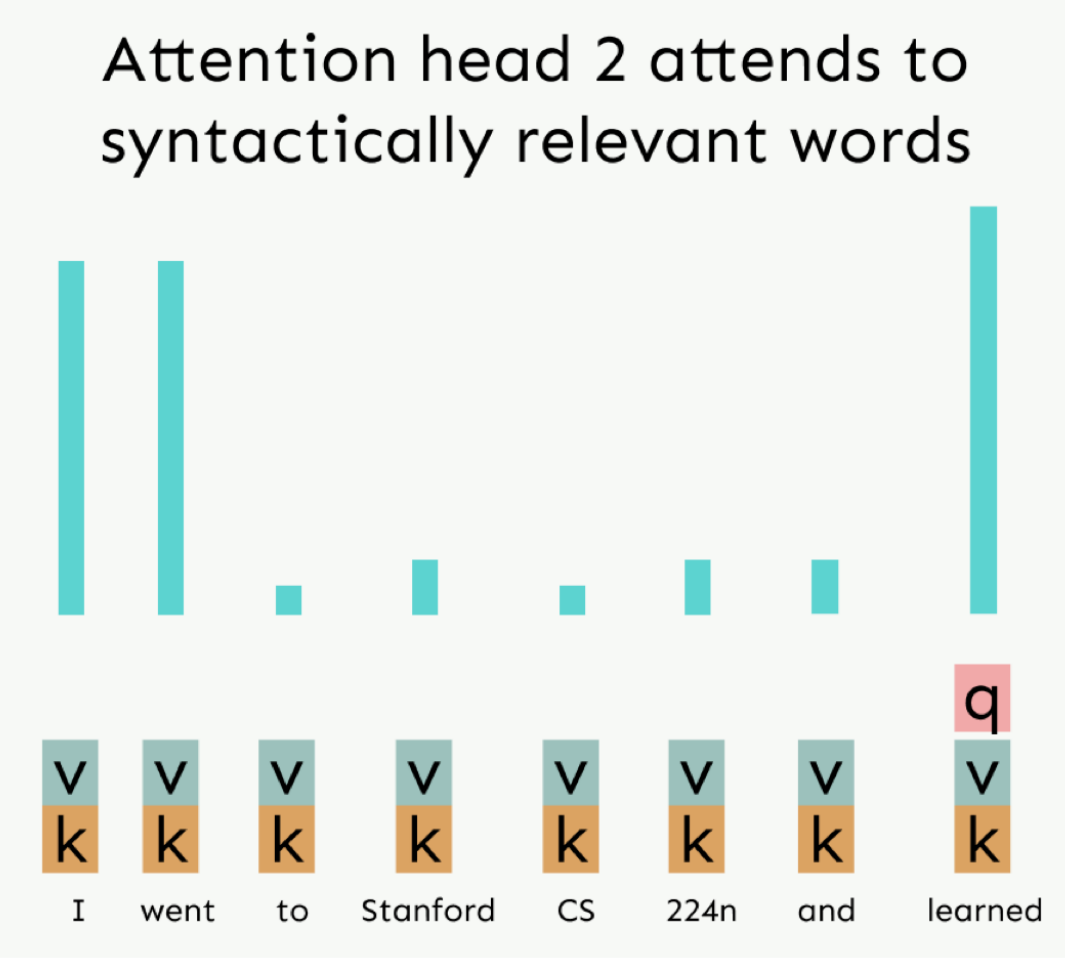


Self-attention

Ramachandran, P., et al. (2019). Stand-Alone Self-Attention in Vision Models. NeurIPS.

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



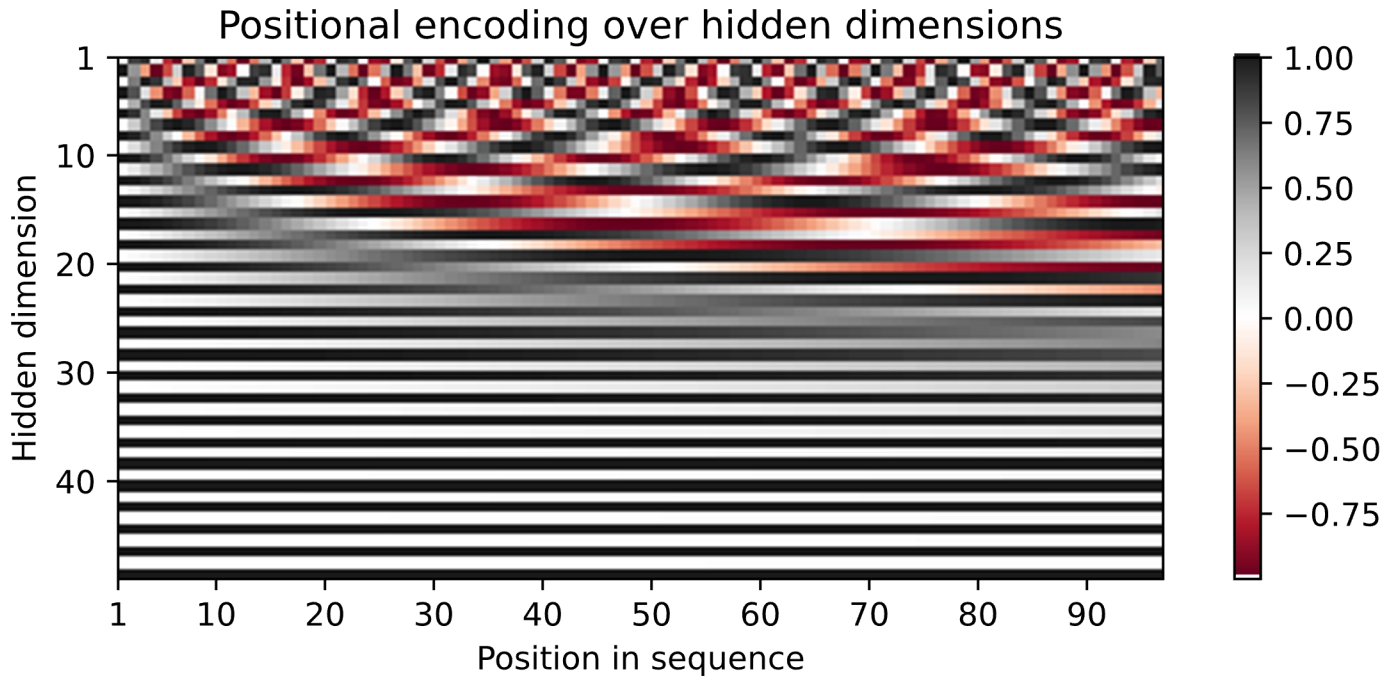


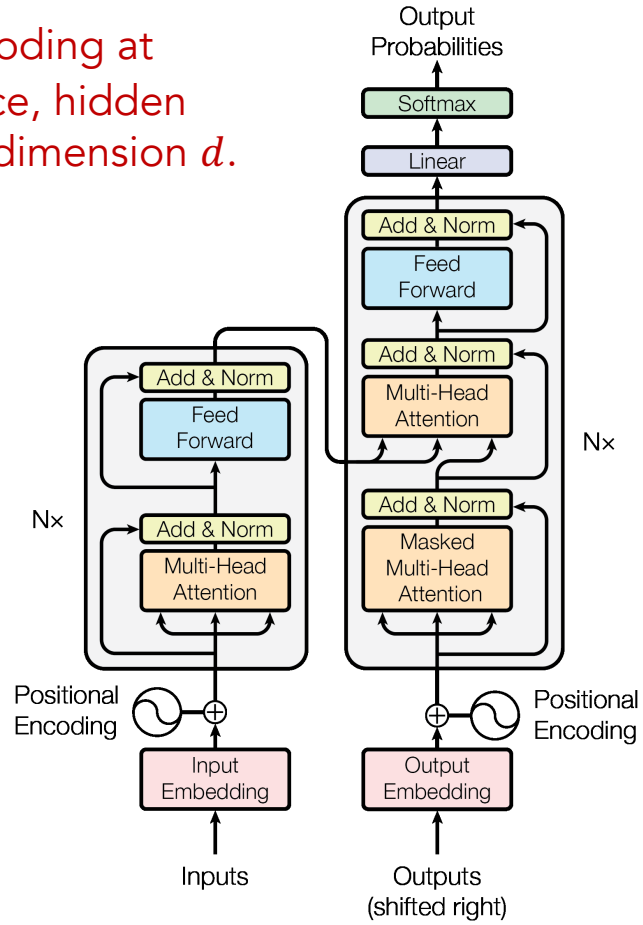
The figure is from the Stanford CS224N course -- https://web.stanford.edu/class/cs224n/

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.

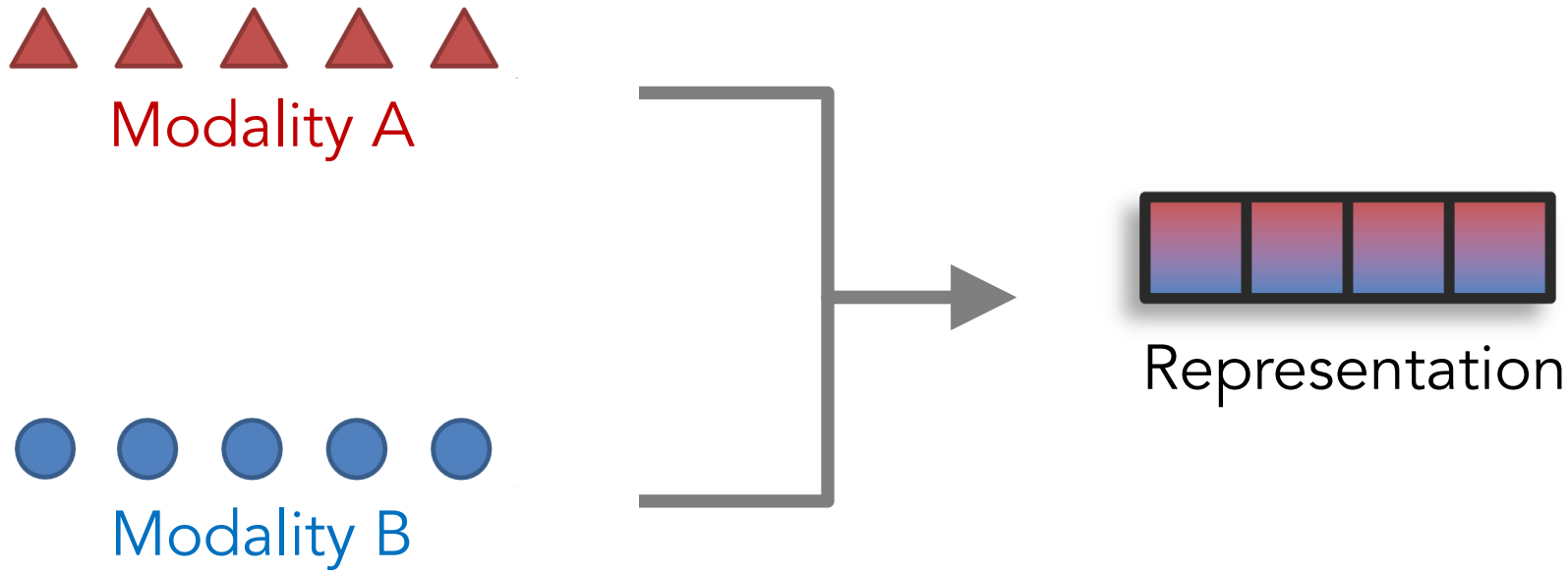
$$PE_{(pos,i)} = \begin{cases} \sin\left(\frac{pos}{10000^{i/d_{\text{model}}}}\right) & \text{if } i \bmod 2 = 0 \\ \cos\left(\frac{pos}{10000^{(i-1)/d_{\text{model}}}}\right) & \text{otherwise} \end{cases}$$

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

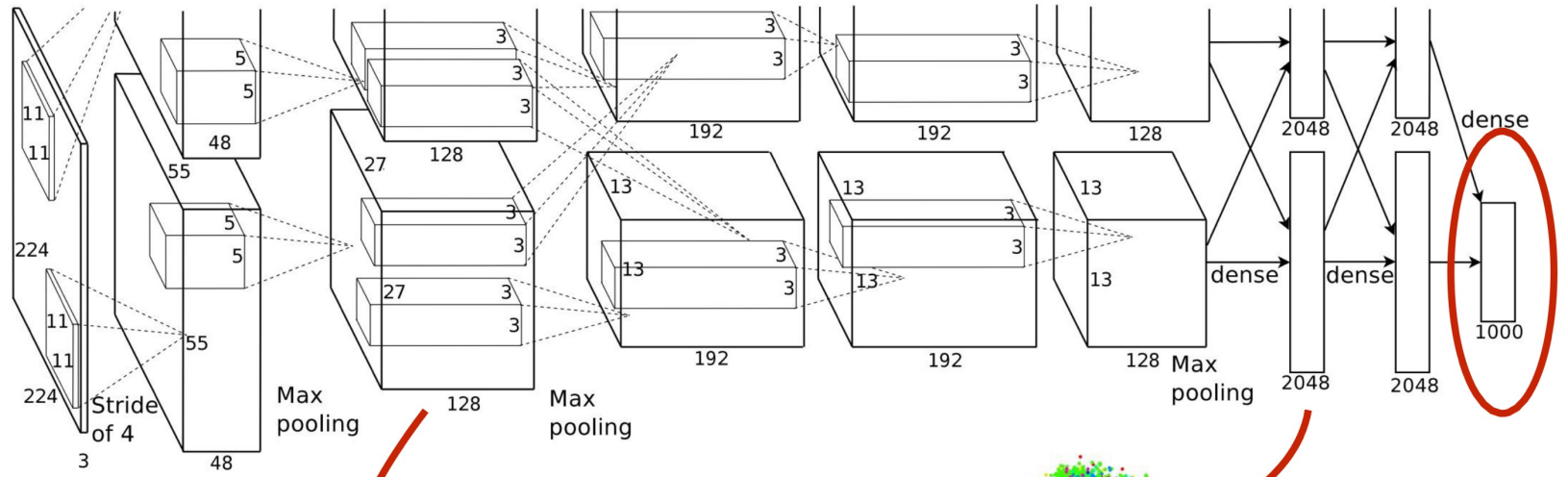




Source -- https://uvadlc-notebooks.readthedocs.io/en/latest/tutorial_notebooks/tutorial6/Transformers_and_MHAttention.html


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.

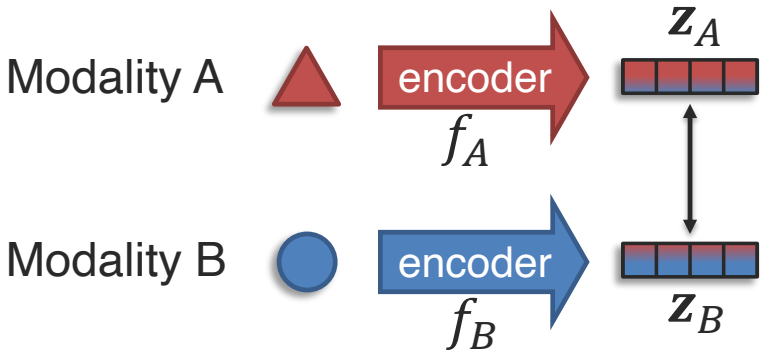


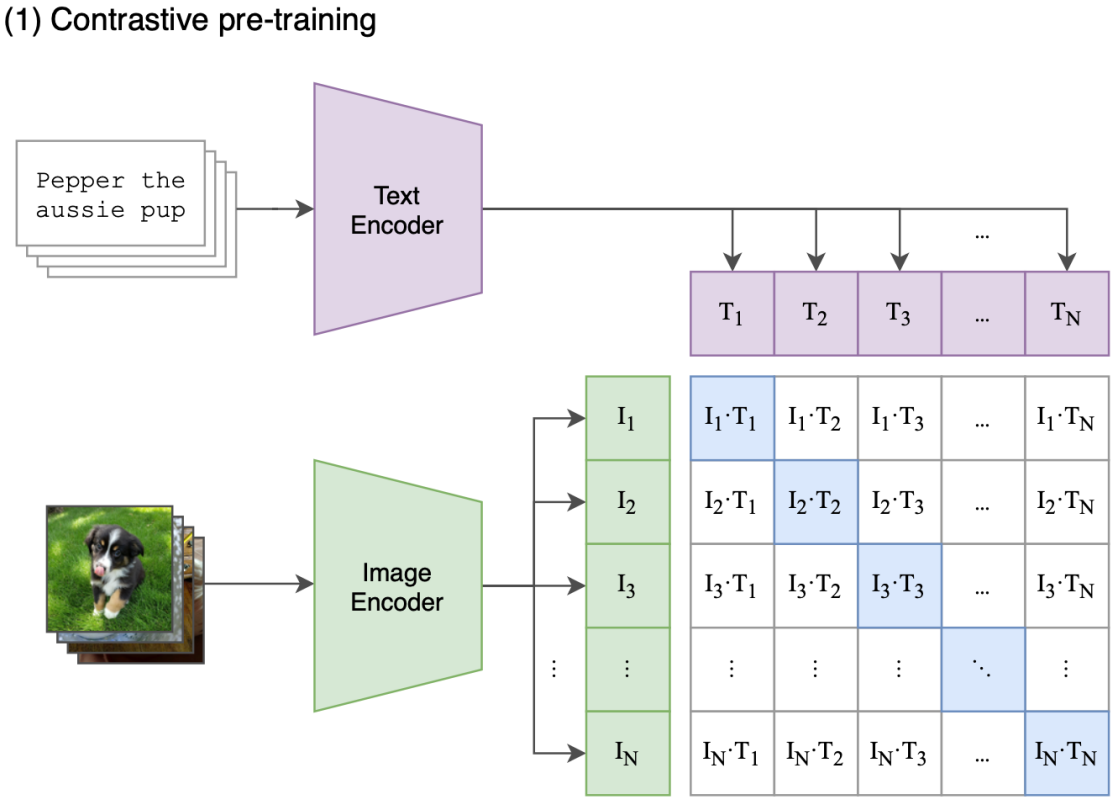
(c) DeCAF$_1$ (d) DeCAF$_6$

Donahue, J., et al. (2014). Decaf: A deep convolutional activation feature for generic visual recognition. ICML.

Source: Cornell CS5670 Computer Vision Course -- https://www.cs.cornell.edu/courses/cs5670/2022sp/lectures/lec21_cnns_for_web.pdf

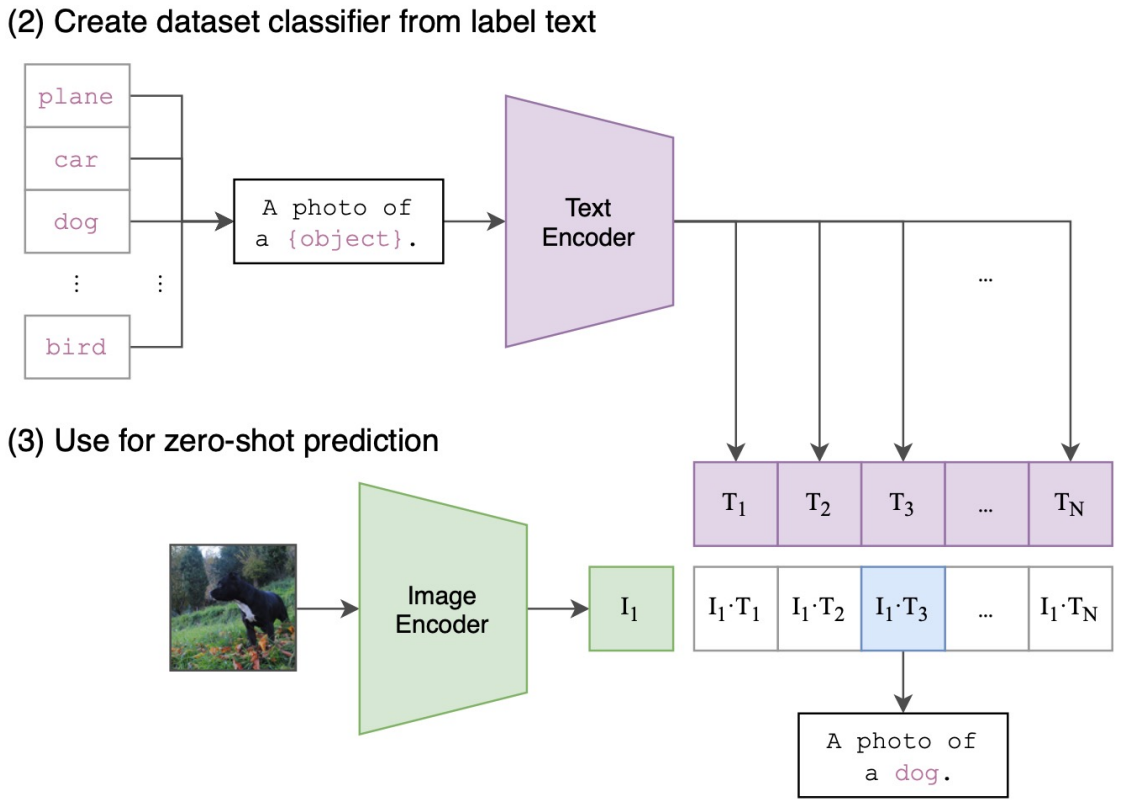
The CLIP model learns a joint text-image representation using a large number of image and text pairs (vision+language→representation).





Radford, A., et al. (2021). Learning Transferable Visual Models From Natural Language Supervision. ICML.

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.

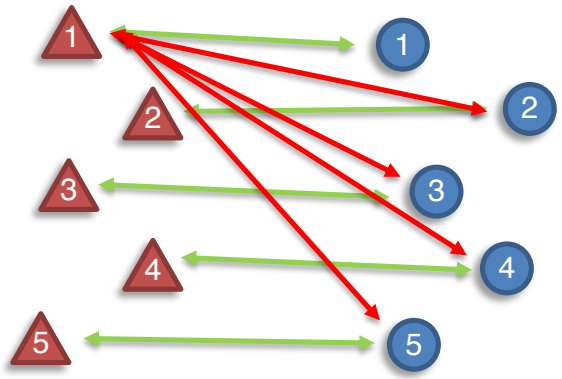


Radford, A., et al. (2021). Learning Transferable Visual Models From Natural Language Supervision. ICML.

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

Paired data: $\{ \blacktriangle, \bullet \}$

(e.g., images and text descriptions)



Simple contrastive loss:

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

positive pairs

negative pair

Similarity functions are often cosine similarity

Popular contrastive loss: InfoNCE

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

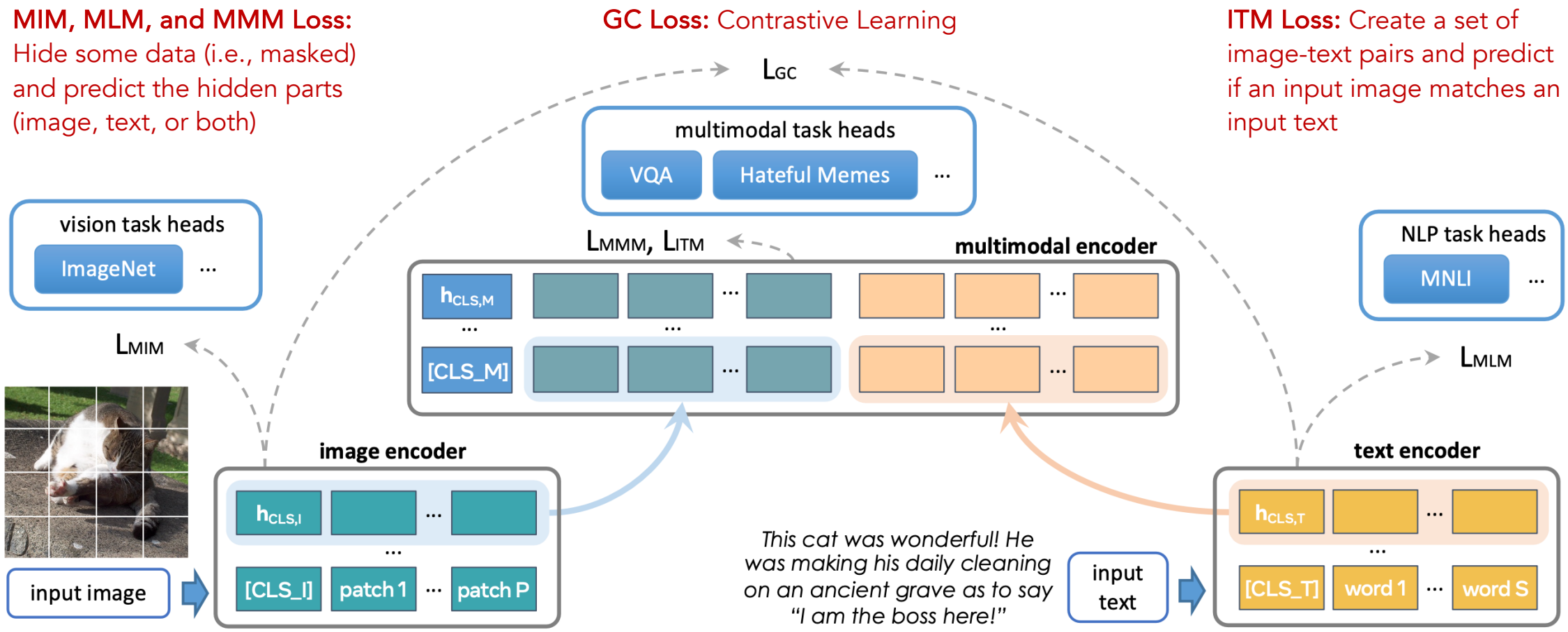
positive pairs

negative pairs and positive pairs

Similarity function can be cosine similarity

Source: CMU 11-777 MMML Course Lecture 3.2 -- https://cmu-multicomp-lab.github.io/mmml-course/fall2023/schedule/

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).



Singh, A., et al. (2022). FLAVA: A Foundational Language And Vision Alignment Model. CVPR.

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?