

ComputerVision

Week9

2025-2

Mobile Systems Engineering
Dankook University

Attention Mechanism in Computer Vision

- Focusing on What Matters Most in an Image

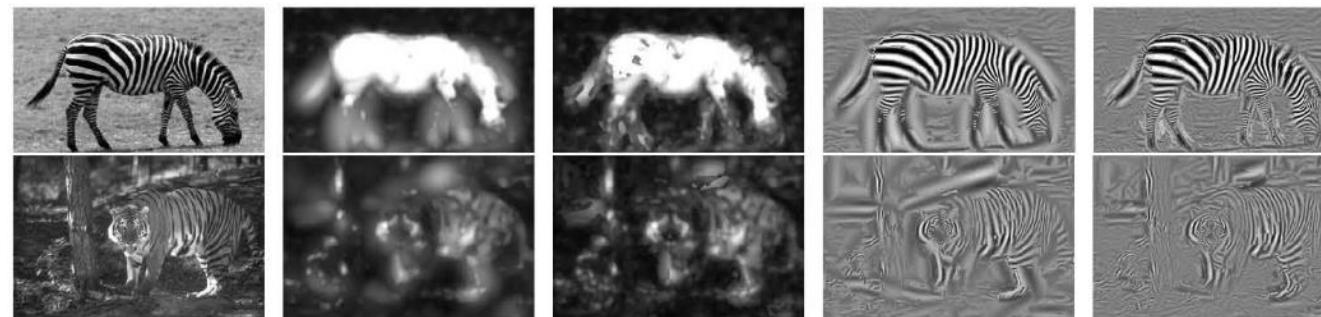
- Before Self-Attention & Transformers...

- We first need to understand the general concept of attention.

- Definition

- Attention = A mechanism that enables models to **focus more on the most relevant parts of the input** while ignoring less important information.

- In Computer Vision, this often means



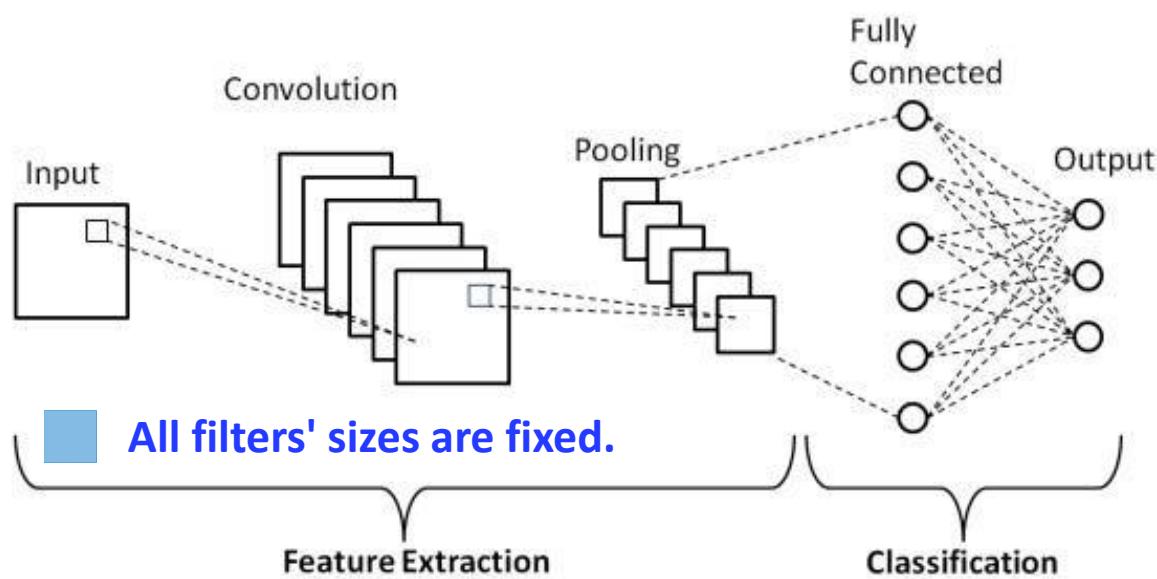
- Prioritizing **important regions** in an image (e.g., objects, edges, or textures)
 - Ignoring irrelevant areas (e.g., plain background)

Attention Mechanism in Computer Vision

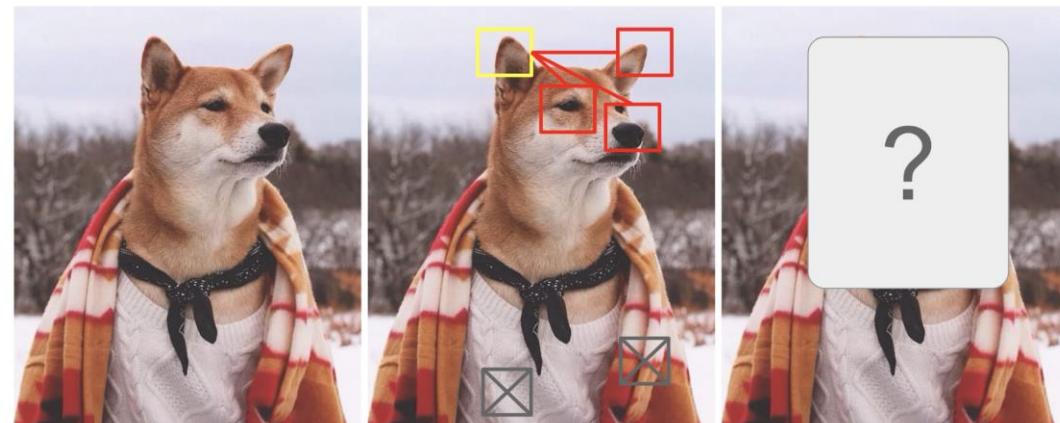
■ Motivation: Why Attention?

- Why “focusing” matters in computer vision models

- 1. The Limitation of Traditional CNNs



We cannot adaptively focus on the image content!



- ✓ **Convolutional kernels** are *fixed-weight filters* applied uniformly across spatial locations.
- ✓ They **cannot adapt** their focus depending on the image content.
- ✓ This can waste computational effort on irrelevant areas and dilute important cues.

Attention Mechanism in Computer Vision

■ Motivation: Why Attention?

- Why “focusing” matters in computer vision models

- 2. Not All Information is Equal

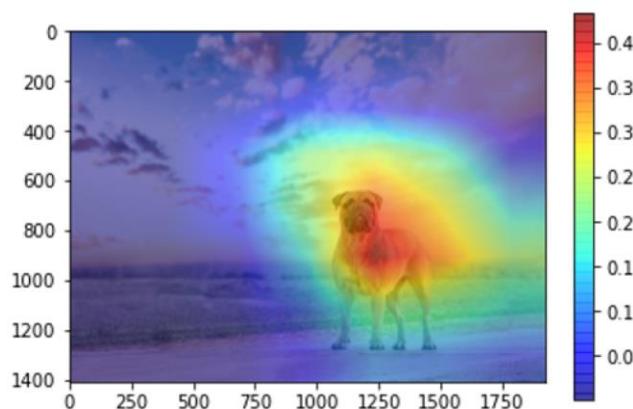
- ✓ In real-world images and videos, **only some regions carry critical information.**

- ✓ Example

- In a crowded street scene, detecting pedestrians and traffic lights matters more than every single background pixel.



(a) Original image



(b) Attention map



Attention Mechanism in Computer Vision

■ Motivation: Why Attention?

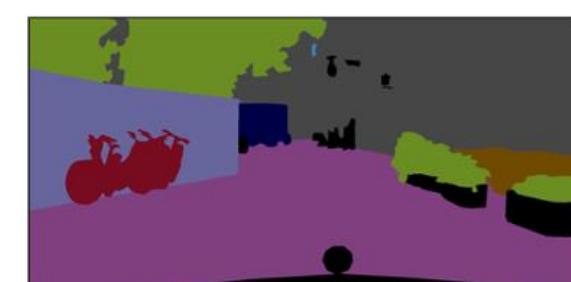
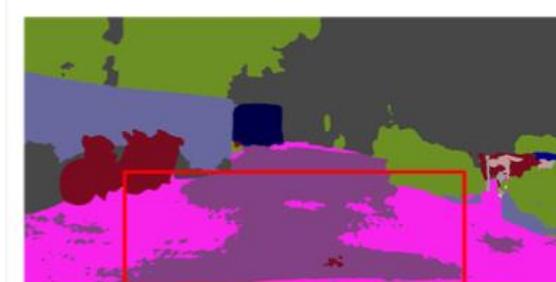
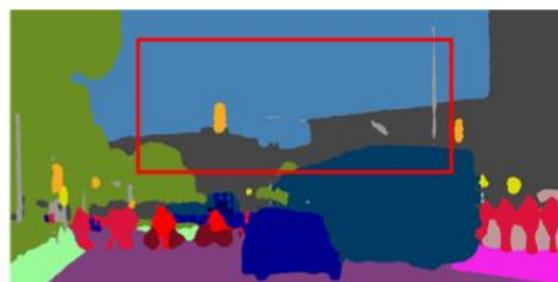
- Why “focusing” matters in computer vision models

- 3. Improves performance in tasks requiring **global context** (scene understanding, semantic segmentation, object detection)

- ✓ **Need for Content-Dependent Processing**

- Some tasks require the model to **dynamically decide where to look** based on the input.
- Analogy to human vision
Our eyes automatically focus on key regions and **ignore unimportant background**, while still being aware of it.

Some semantic segmentation issues observed in the Cityscapes dataset.



Case1. Missing small and thin objects

Case2. Incomplete segmentation of large objects (background)

Motivation: Why Attention?

- Why “focusing” matters in computer vision models

- Key Takeaway for Computer Vision

- 1. CNN → fixed & local
 - 2. Attention → adaptive & global
 - 3. Attention allows the model to
 - ✓ Allocate **more capacity** to important areas.
 - ✓ Capture **long-range dependencies** in a single step.
 - ✓ Improve interpretability with attention maps.

Early Attention in AI

■ Early Attention in Artificial Intelligence

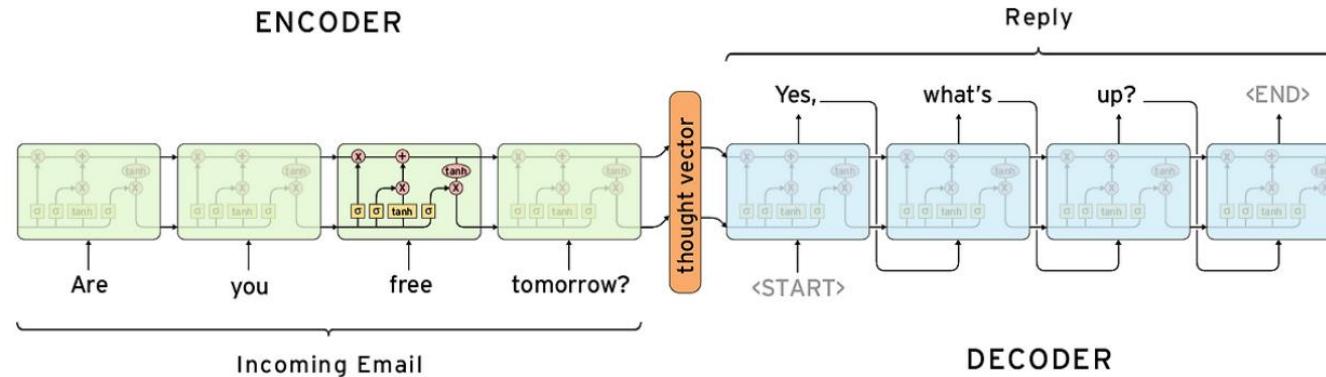
- Attention mechanisms were first introduced in the field of **Neural Machine Translation (NMT)** around 2014–2015 (Bahdanau et al., 2014; Luong et al., 2015).
- **Motivation in NMT**
 - Seq2Seq models encode an input sentence into a **fixed-length vector** before decoding.
 - Problem: When the input sequence is long, **information loss** occurs (compression bottleneck).
 - RNN-based decoders also suffered from **vanishing gradients**, making it hard to learn long-range dependencies.
- **Core Idea**

Allow the decoder to **dynamically attend** to different parts of the input sequence for each output word.

Early Attention in AI

■ Fixed-Length Bottleneck Problem

- Seq2Seq model without attention.



- Encoder → compresses all input tokens into a single vector c .
- Decoder → uses only c to generate all output tokens.
- Issues
 - ✓ 1. Loss of fine-grained details for long sequences.
 - ✓ 2. Uniform treatment of all tokens regardless of relevance.
- Example
 - ✓ *Summarizing a whole book into one sentence, then trying to answer detailed questions about it — impossible to recall all specific details.*

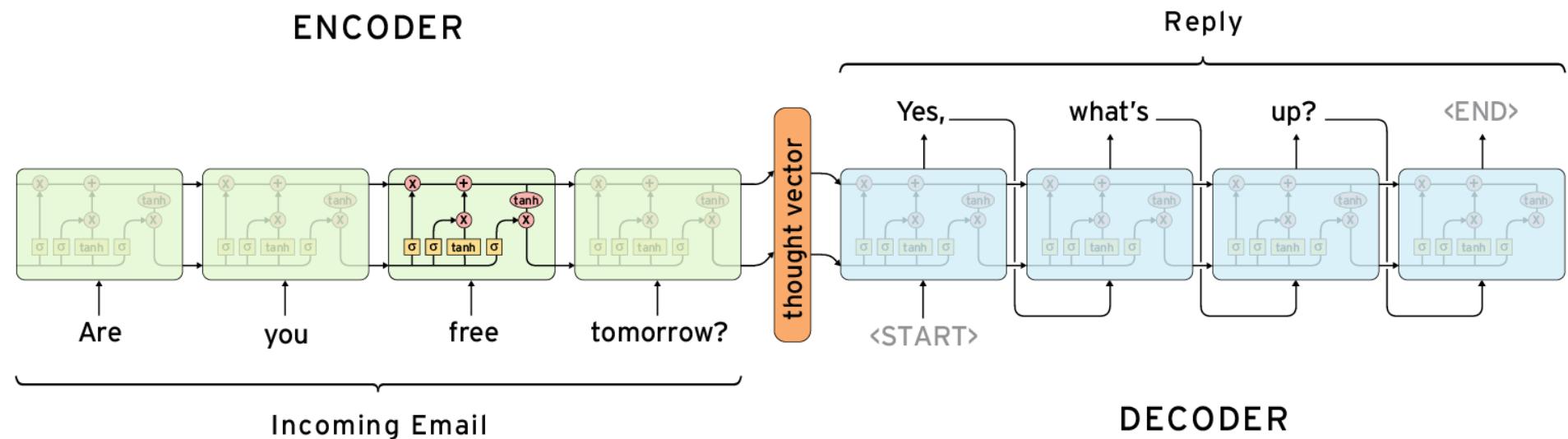
Early Attention in AI

■ What is Seq2Seq?

- **Definition**

- **Seq2Seq (Sequence-to-Sequence)** models are deep learning architectures that transform one sequence into another.
- Mostly used in **Natural Language Processing (NLP)** tasks such as machine translation, text summarization, and question answering.

- **Structure**



- Two main components

- ✓ **Encoder** — processes the input sequence and produces a fixed-length **context vector**.

- ✓ **Decoder** — generates the output sequence using the context vector.

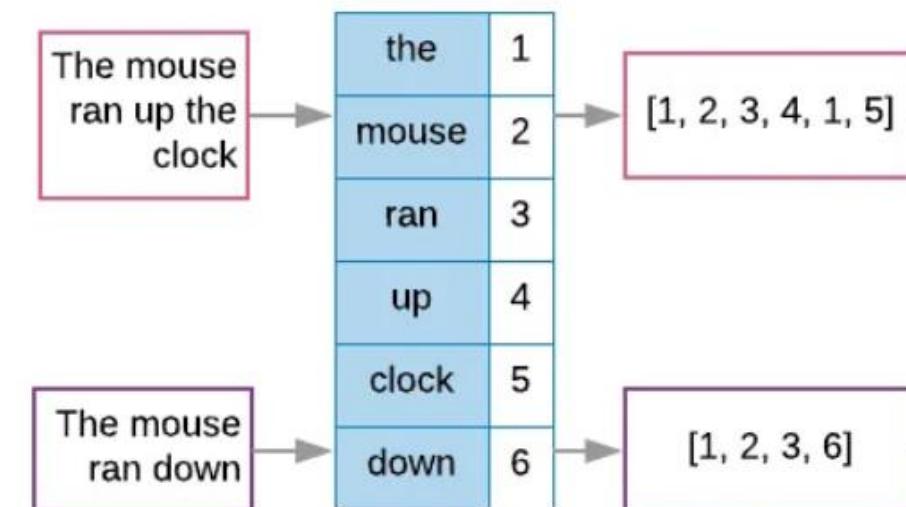
Early Attention in AI

■ How the Encoder Works

- The **encoder** transforms an input sequence (e.g., a sentence) into a compact representation that summarizes its meaning.
 - We can break this process into **three main stages**.

• Step 1. Tokenization

- **Goal:** Break down a long text string into smaller units called **tokens**.
- Tokens can be **words**, **subwords**, or **characters** depending on the task.
- Example
 - ✓ "Hello, World!" → ["Hello", ",", "World", "!"]



- Each token represents a **basic unit of meaning**.
 - ✓ Tokenization converts **unstructured text** into a **structured list of symbols**.

Early Attention in AI

■ How the Encoder Works

• Step 2. Vectorization

- Goal: Convert each token into a **numeric vector** that the model can understand or process.

✓ This is called **word embedding**.

- Each token is mapped to a **fixed-length dense vector** capturing

✓ **Semantic meaning** (similar words have similar vectors)

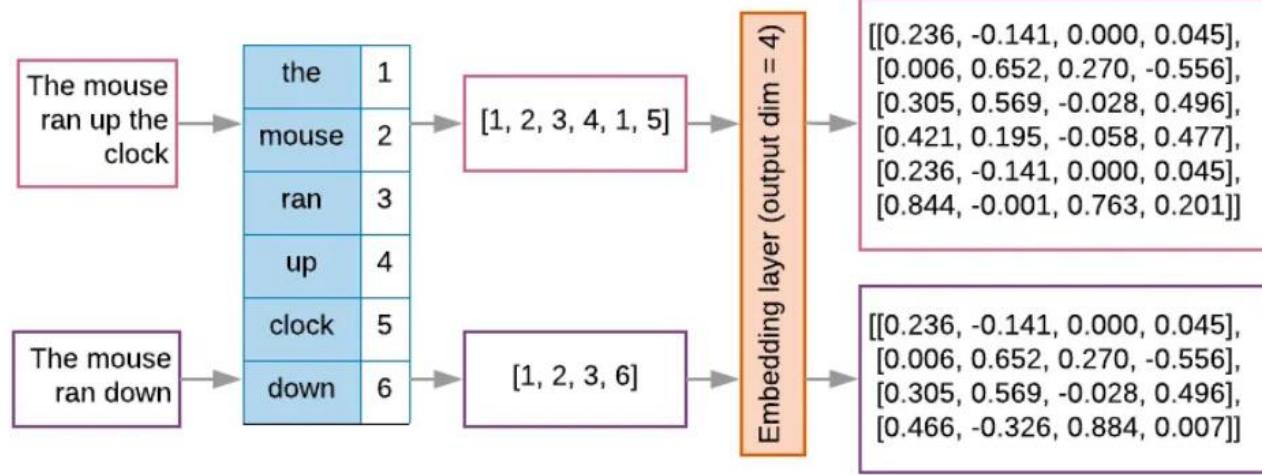
✓ **Syntactic roles** (e.g., verbs vs. nouns)

- Examples of embedding techniques

✓ **Word2Vec, GloVe, FastText**

- Benefit

✓ High-dimensional text data is transformed into **lower-dimensional, computationally efficient representations**.



Early Attention in AI

■ How the Encoder Works

• 3. Context Vector Generation

- Goal: Summarize the entire input sequence into a **single fixed-length vector** (the **context vector**).

- ✓ The encoder processes embeddings **sequentially** using **RNN, LSTM, or GRU**.

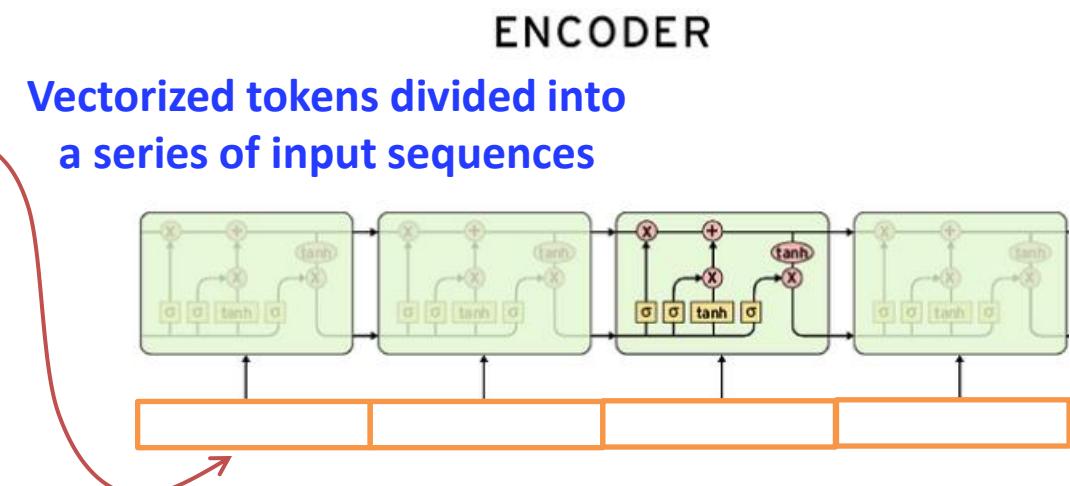
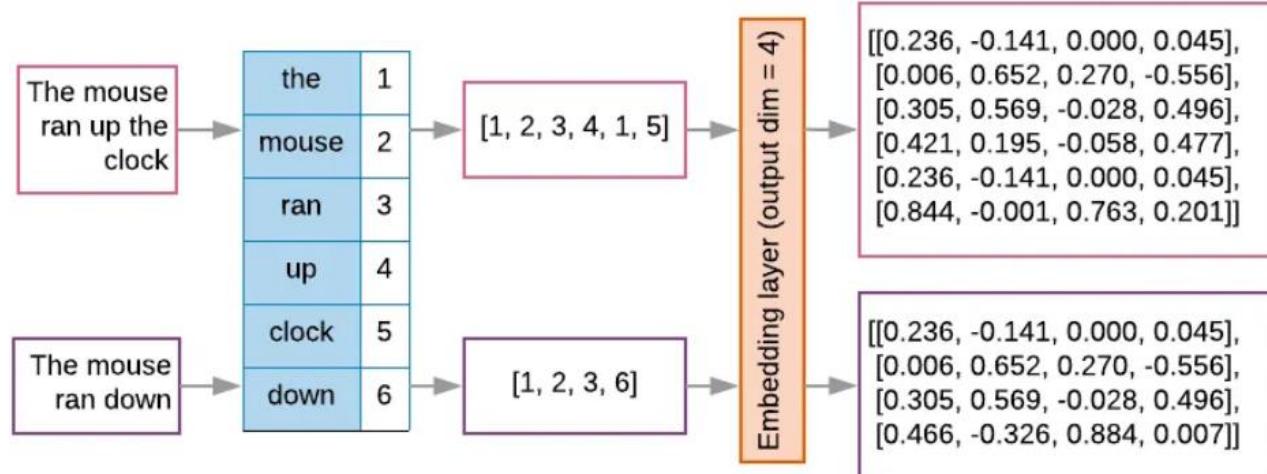
- ✓ At each step, the model updates a **hidden state**.

- ✓ The **final hidden state** represents the entire input — this is the **context vector**.

- Limitation

- ✓ Compressing long sequences into a single vector can lead to **information loss**.

- ✓ This **bottleneck** is one of the main reasons why the **attention mechanism** was introduced.



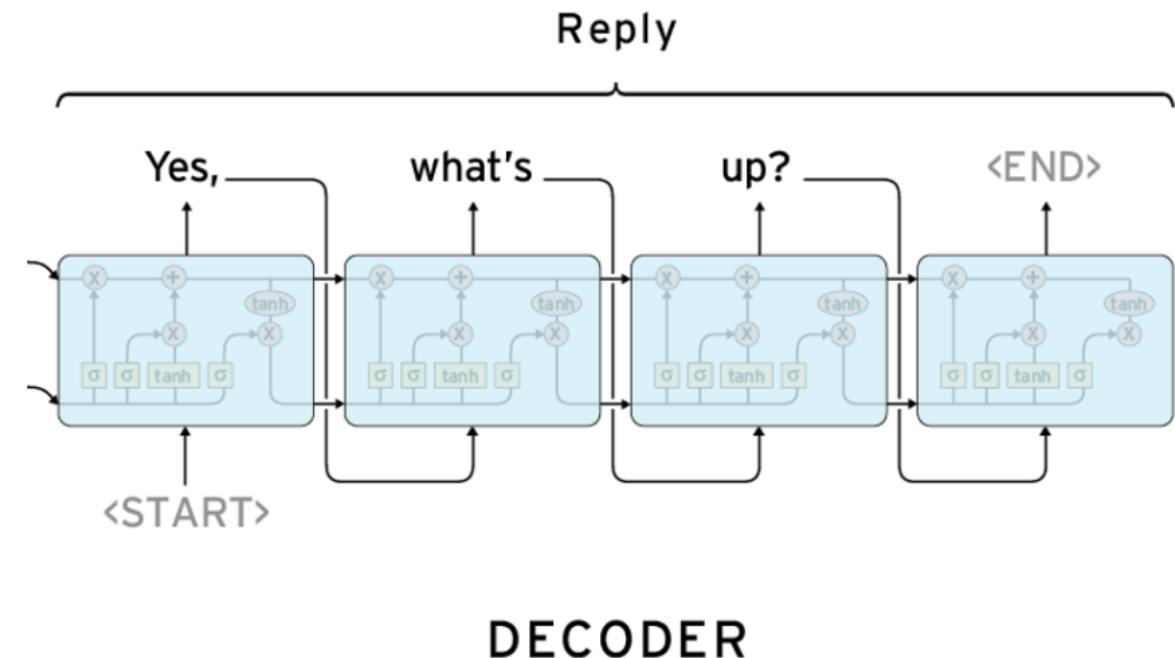
Early Attention in AI

■ How the Decoder Works

- The decoder takes the **context vector** and **hidden state** from the encoder, and generates the **output sequence** step-by-step.
It works in a **recursive** manner until the complete output is formed.

- **Step-by-Step Sequence Generation**

- **Input to the decoder**
 - ✓ **Context vector** from the encoder
 - ✓ **Hidden state** (summarizing the encoded sequence)
 - ✓ **Previous output token** (for all steps except the first)
- The decoder predicts **one token at a time**.
- Each predicted token becomes **input for the next step**.



Early Attention in AI

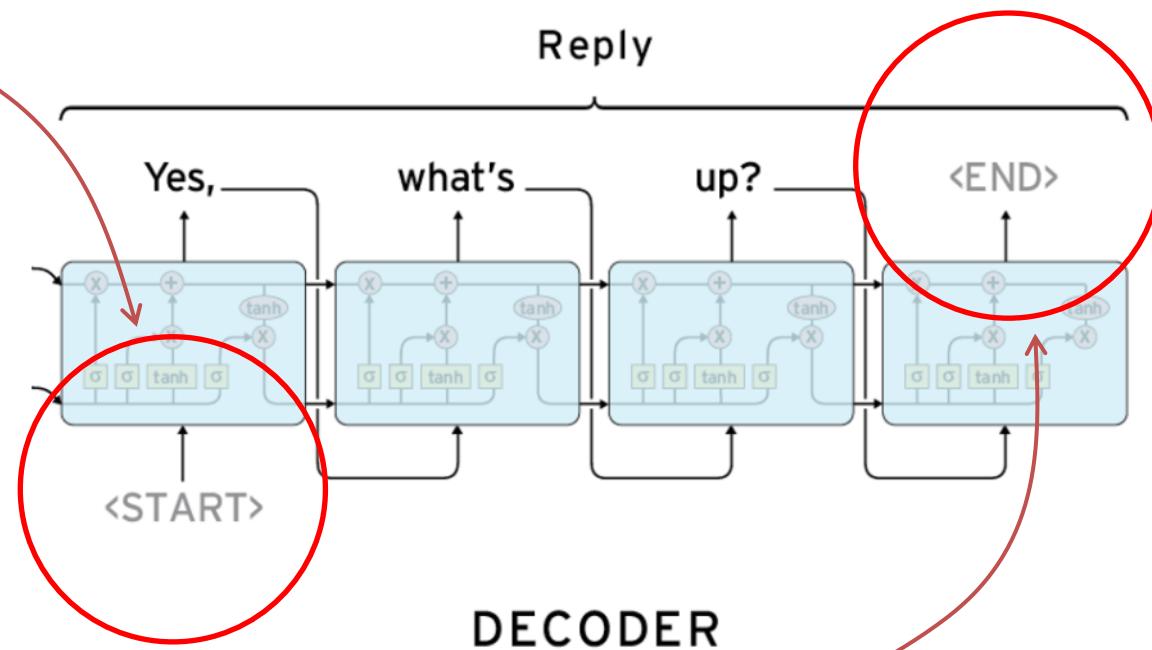
■ How the Decoder Works

- Special Start Token (<START>)

- At the **first step**, there is no previous output token.
 - ✓ Instead, a special **start token** (e.g., <START>, <GO>, <s>) is fed into the decoder.

- Recurrent Prediction Process

- At each decoding step
 - ✓ Combine **previous token embedding, context vector, and hidden state**.
 - ✓ Pass through an RNN/LSTM/GRU cell.
 - ✓ Apply **softmax** to produce a probability distribution over possible next tokens.
 - ✓ Choose the token with the **highest probability**.
- Continue until a special **end token** (<END>) is produced.



Early Attention in AI

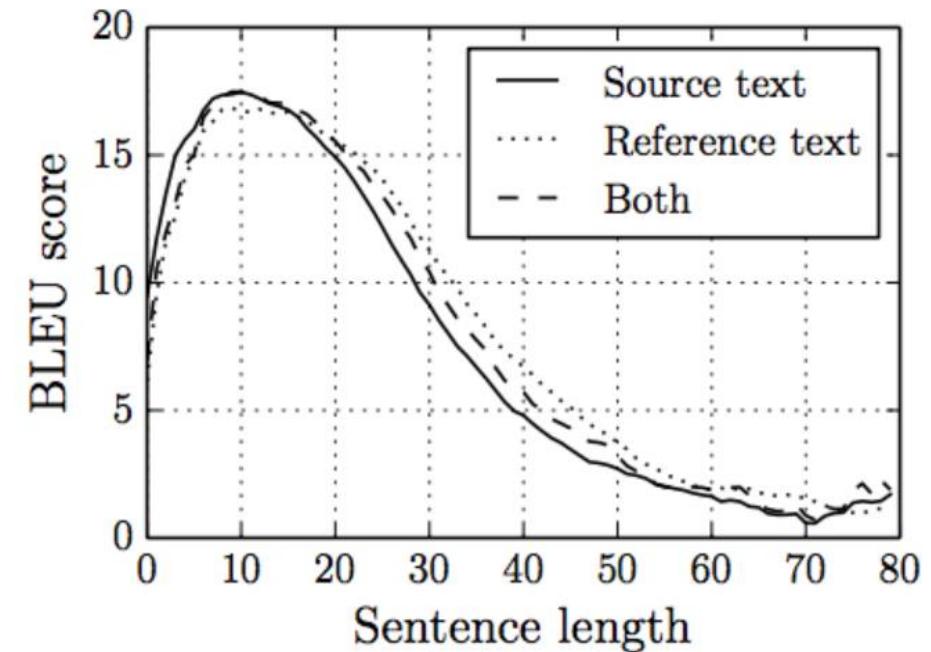
■ Key Limitation of Fixed-Length Context Vector in Seq2Seq Models

- Key Limitations

- In traditional Seq2Seq models, the decoder **relies entirely on a single fixed-length context vector** from the encoder.
- This design **cannot effectively retain information for long sequences**.
- As the entire sequence is compressed into one vector, the model tends to **forget early parts of the input sequence** when processing long inputs.

- Evidence – BLEU Score Degradation

- BLEU (Bilingual Evaluation Understudy) score measures similarity between generated translations and reference translations.
- Graph shows
 - ✓ For **short sentences**, Seq2Seq works well (high BLEU scores).
 - ✓ For **long sentences**, BLEU score drops sharply
→ indicating loss of information from earlier tokens.



Early Attention in AI

■ Why Attention Became Necessary

- In traditional Seq2Seq
 - The decoder relies **only** on the single context vector.
 - Cannot “look back” at specific parts of the input sequence once encoding is done.
- **Attention Mechanism** solves this by
 - Allowing the decoder to **access all encoder hidden states** at each decoding step.
 - Assigning **different weights** to different parts of the input depending on relevance.
- This enables
 - Better handling of **long sequences**.
 - More accurate alignment between input and output.

Early Attention in AI

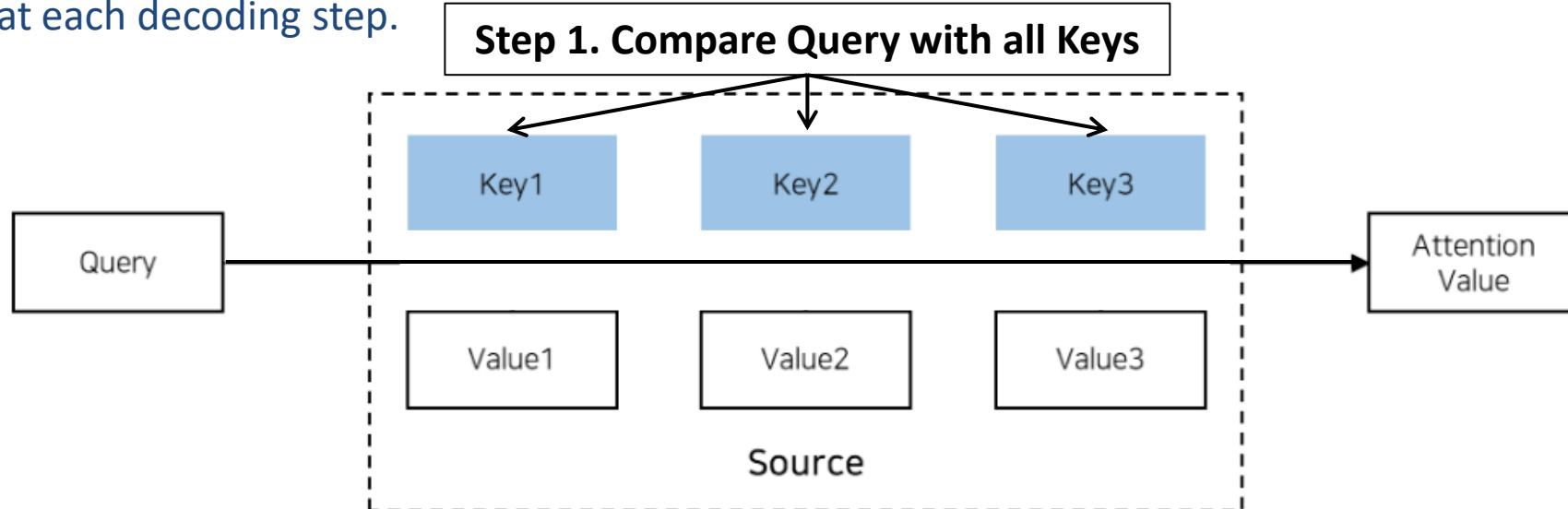
■ Attention as a Solution

- From a single bottleneck vector to step-wise, adaptive context retrieval

- Key Idea

- ✓ Instead of relying on **one fixed-length context vector**, the decoder can dynamically **refer back to all encoder hidden states** at each decoding step.

- How it works



- ✓ **Step 1. Compare Query with all Keys**

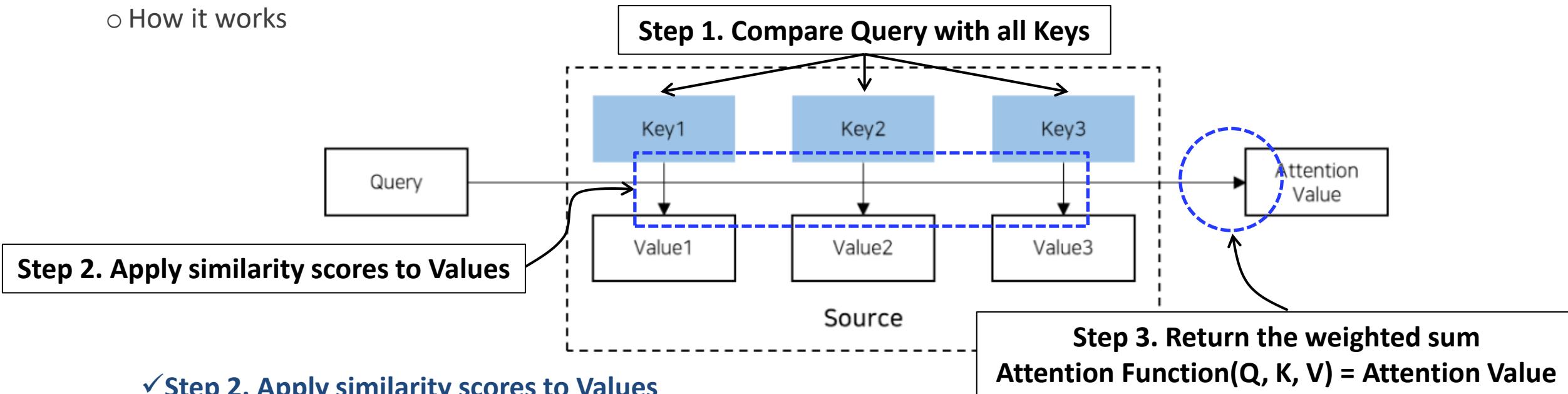
- **Query (Q)**: the decoder's current hidden state at time step t .
 - **Keys (K)**: all encoder hidden states representing each input token.
 - Compute similarity scores between Q and every K.

Early Attention in AI

■ Attention as a Solution

- From a single bottleneck vector to step-wise, adaptive context retrieval

- How it works



✓ Step 2. Apply similarity scores to Values

- **Values (V)**: vectors associated with each Key (often the encoder hidden states themselves).
- Convert similarity scores into **weights** (e.g., via softmax) and multiply each Value by its corresponding weight.

✓ Step 3. Return the weighted sum

- Add up the weighted Values to produce the **Attention Value** (context vector) for step t .
- This context is **tailored** to what the decoder needs right now.

Early Attention in AI

■ Attention Mechanism — Step-by-Step

- Goal

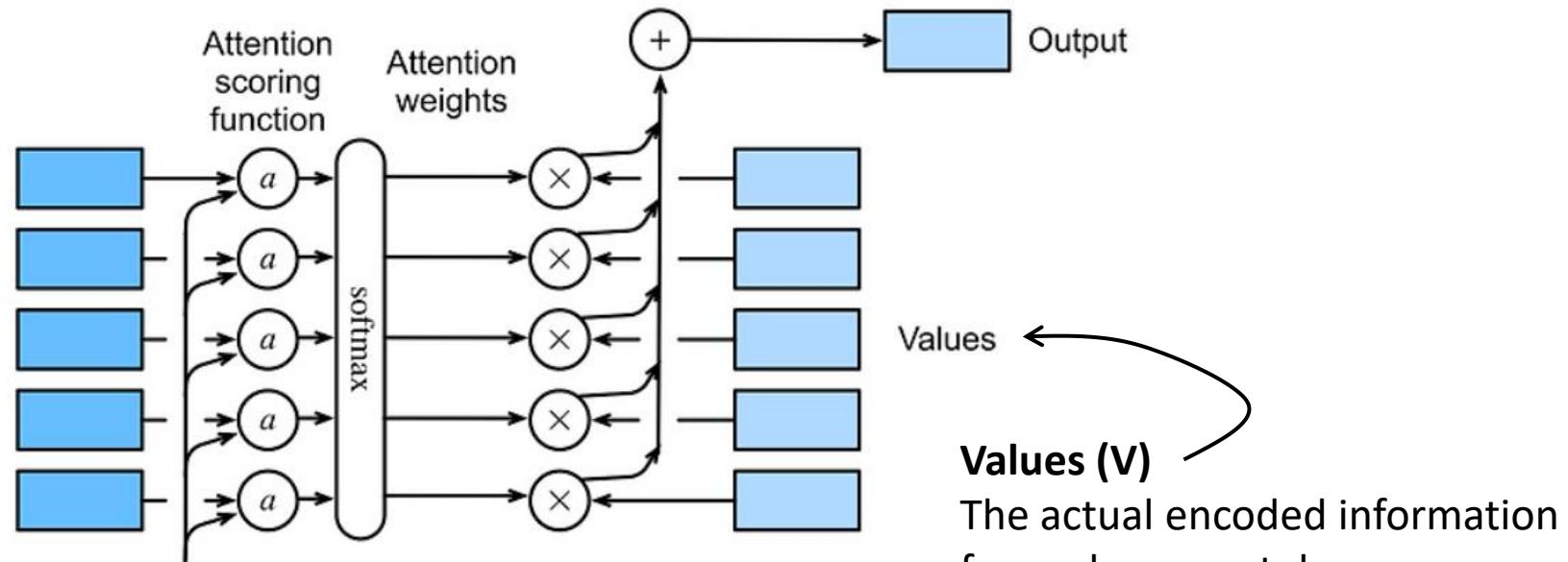
- Enable the decoder to focus on the most relevant parts of the encoder's output when generating each token.

- Step 1: Inputs

Keys (K)
All encoder hidden states — act like “labels” for each source token.

Query (Q)

The current decoder hidden state at time step t — represents “what we’re looking for” at this point in decoding.



Values (V)

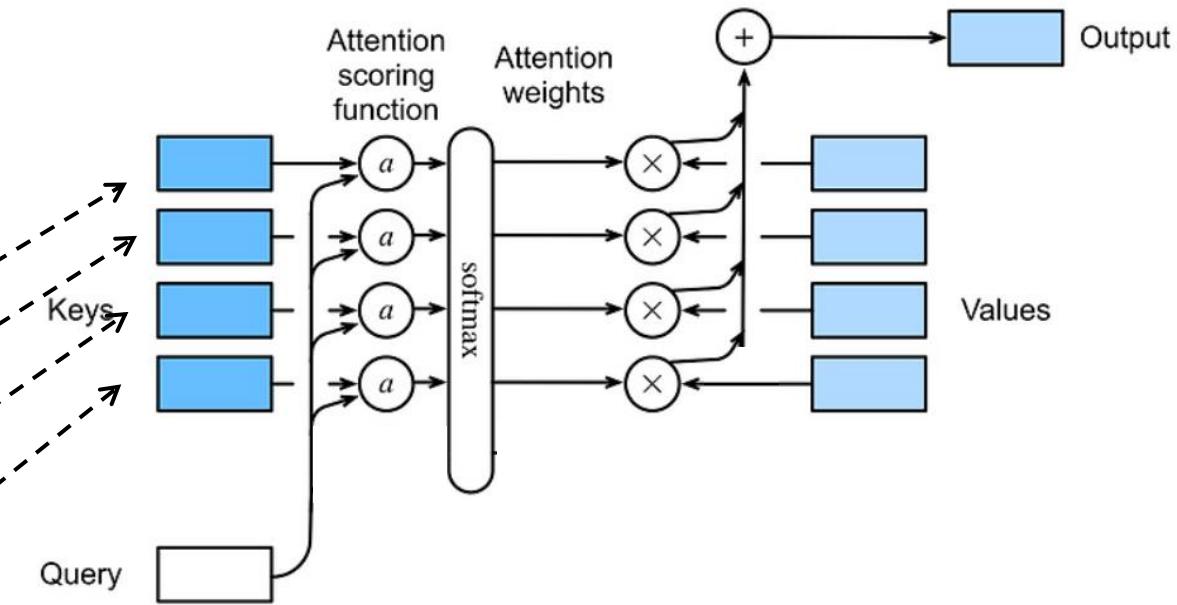
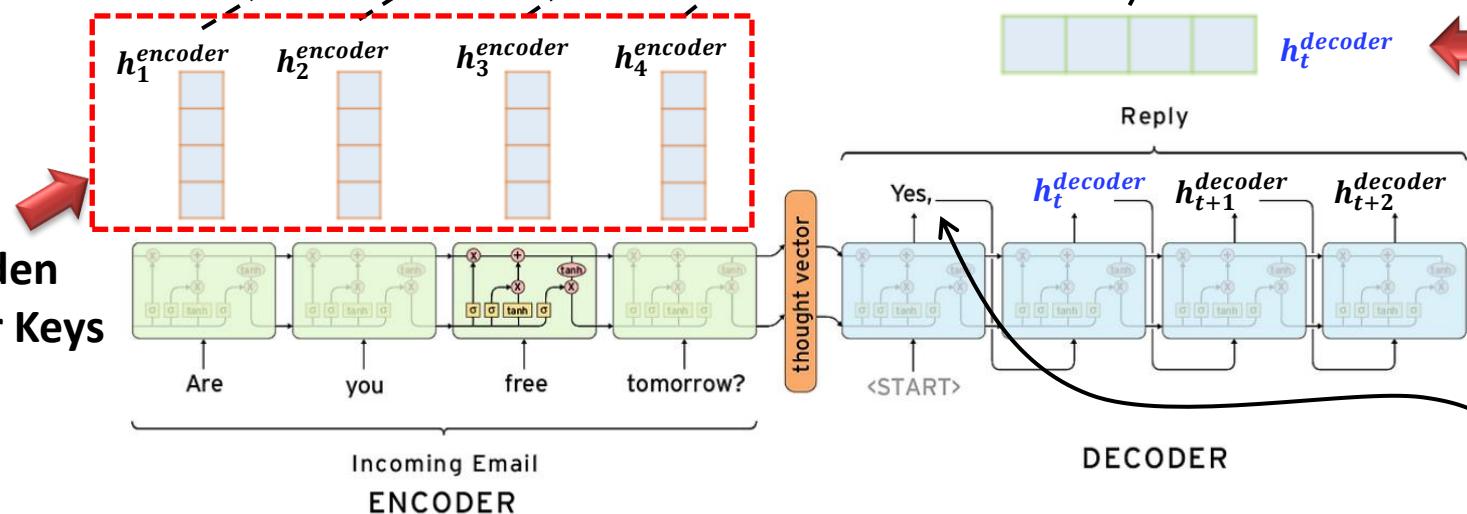
The actual encoded information for each source token.

Early Attention in AI

■ Attention Mechanism — Step-by-Step

• Step 2: Compute Similarity (Attention Scores)

- Compute Similarity (Attention Scores)
- Compare the **Query** with each **Key** to measure similarity.
- Common similarity measure: **Dot Product**
- ✓ $score(q, k_i) = q \cdot k_i$
- Result
 - ✓ A set of raw scores indicating how much attention each source token should receive.



Early Attention in AI

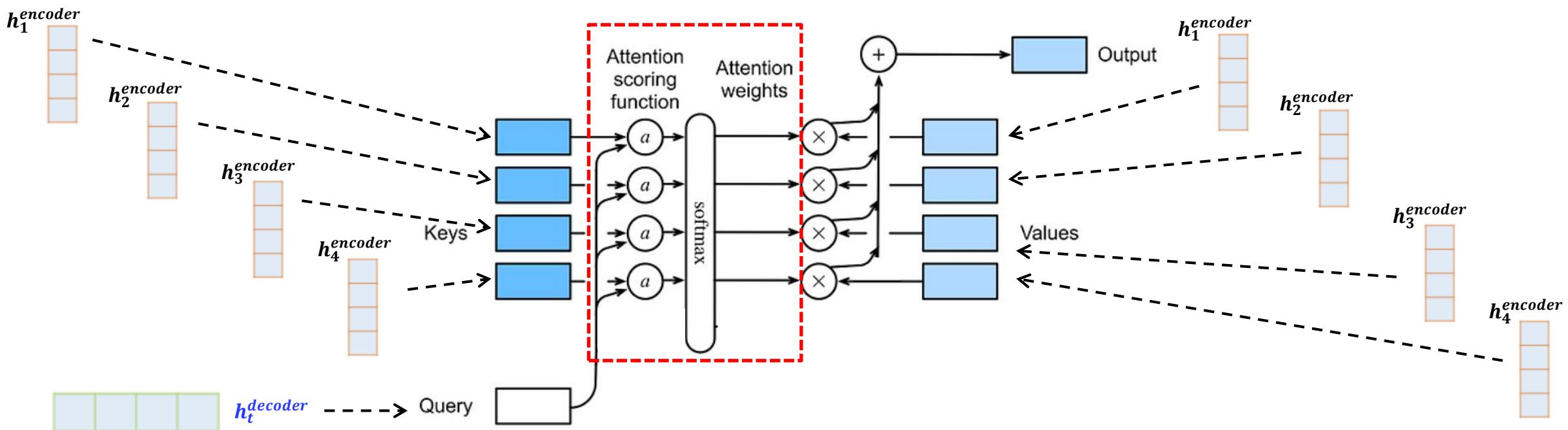
■ Attention Mechanism — Step-by-Step

- Step 3: Normalize (Attention Weights)

- Apply softmax to scores so they sum to 1:

$$\checkmark \alpha_i = \frac{\exp(score(q, k_i))}{\sum_{j=1}^n \exp(score(q, k_j))}$$

- Produces attention weights $\alpha_1, \alpha_2, \dots, \alpha_n$ indicating relative importance of each token.



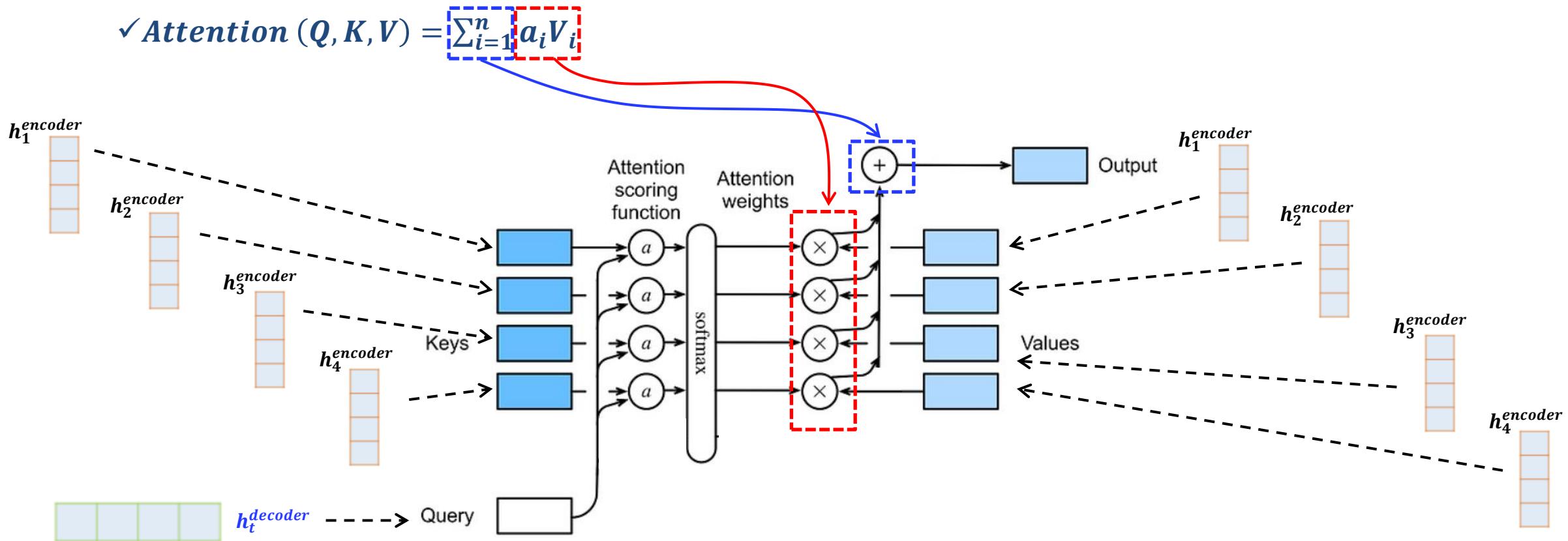
Early Attention in AI

■ Attention Mechanism — Step-by-Step

- Step 4: Weighted Sum

- Multiply each **Value** vector by its corresponding attention weight.
- Sum the results to obtain the **Attention Value**

$$\checkmark \text{Attention } (Q, K, V) = \sum_{i=1}^n a_i V_i$$

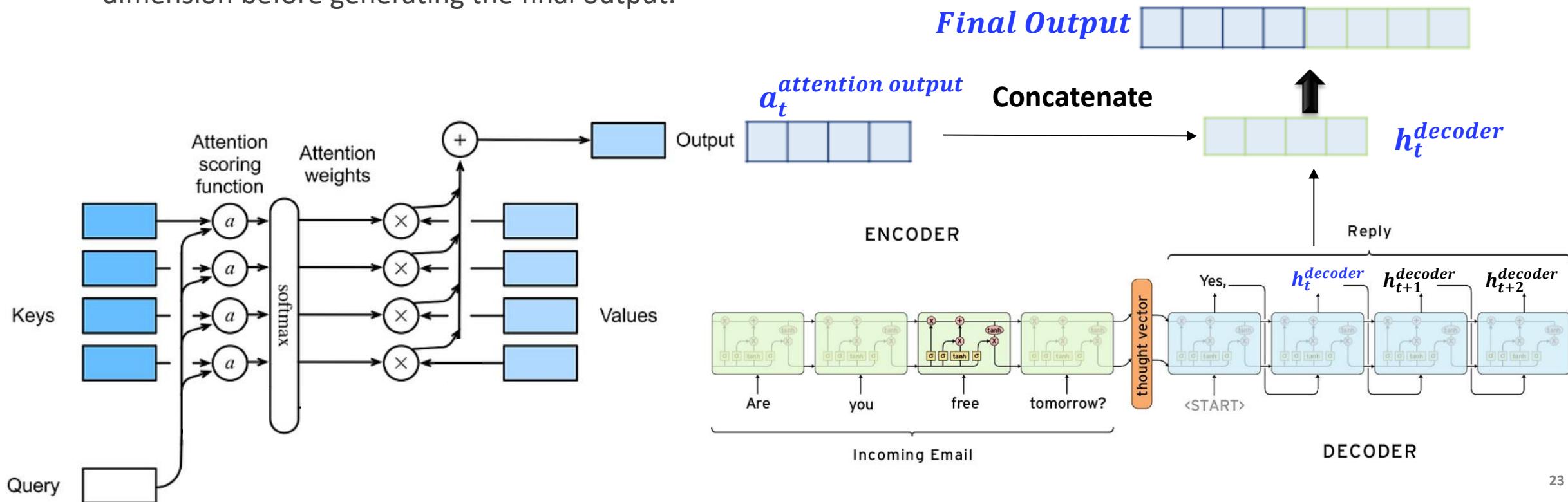


Early Attention in AI

■ Attention Mechanism — Step-by-Step

- **Step 5: Output to Decoder**

- The Attention Value is passed to the decoder along with its own state.
- Allows the decoder to incorporate **context-specific information** for the current output token.
- **Note:** The combined vector can be multiplied by a specific weight matrix to project it back to the original output dimension before generating the final output.

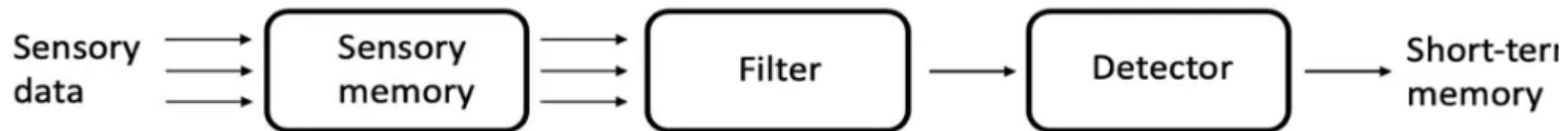


Attention in Computer Vision

■ Origins of Attention in Cognitive Psychology

- In cognitive psychology, attention is defined as a **cognitive process that selectively focuses on the most important parts of information.**
 - Human visual attention operates as a combination of **low-level visual features** and **high-level cognitive guidance**.

• Broadbent's Early Selection Model



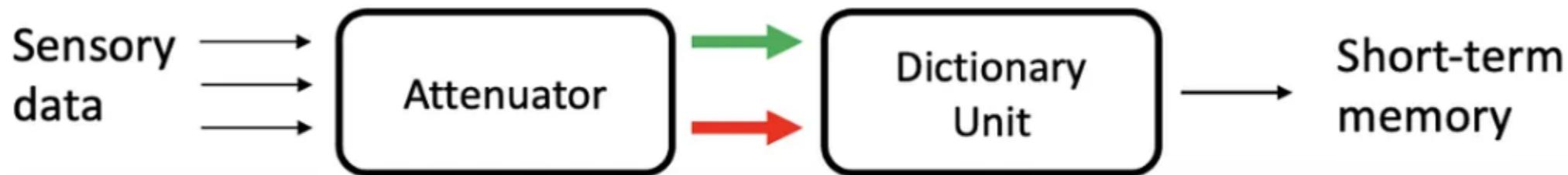
Broadbent's model of cognitive attention.

- Sensory data is first stored briefly in **sensory memory** and then filtered based on physical features (e.g., pitch of sound, color, shape).
- The **filter** selects only the relevant information, which is then passed to the **detector** and stored in **short-term memory**.
- Selection is based primarily on low-level physical characteristics, and higher-level semantic information is processed later.

Attention in Computer Vision

- Origins of Attention in Cognitive Psychology

- Treisman's Late Selection Model



Treisman's model of cognitive attention.

- Incoming data is not completely blocked but rather attenuated based on importance through an **attenuator**.
- Highly important data (green arrow) is passed strongly, while less important data (red arrow) is passed weakly. The **dictionary unit** determines whether the information is activated based on meaning and context.
- For example, one's own name can be activated with minimal signal strength, while irrelevant words require much higher activation levels.

Attention in Computer Vision

■ Origins of Attention in Cognitive Psychology

- **Role in Artificial Intelligence (Deep Learning)**

- The human **selective attention mechanism** has inspired its integration into deep learning, where it is used to **assign higher weights to important parts of the input data**.
- **Role:** Suppress irrelevant information and emphasize important information, enabling more efficient learning.
- **Effect:** Reduces information loss and improves performance, particularly for long input sequences or complex visual data.

- **Role in Computer Vision**

- **1. Concept & Role**

- ✓ In CV, attention is a **dynamic weight adjustment process** applied to feature maps between convolution layers.
 - ✓ **Purpose:** Guides the model to focus on the **most relevant features** for the task.
 - ✓ **Applications:** Almost tasks

- Image Classification · Object Detection · Semantic Segmentation · Video Understanding · Image Generation · 3D Vision · Multi-modal Tasks · Self-supervised Learning

Attention in Computer Vision

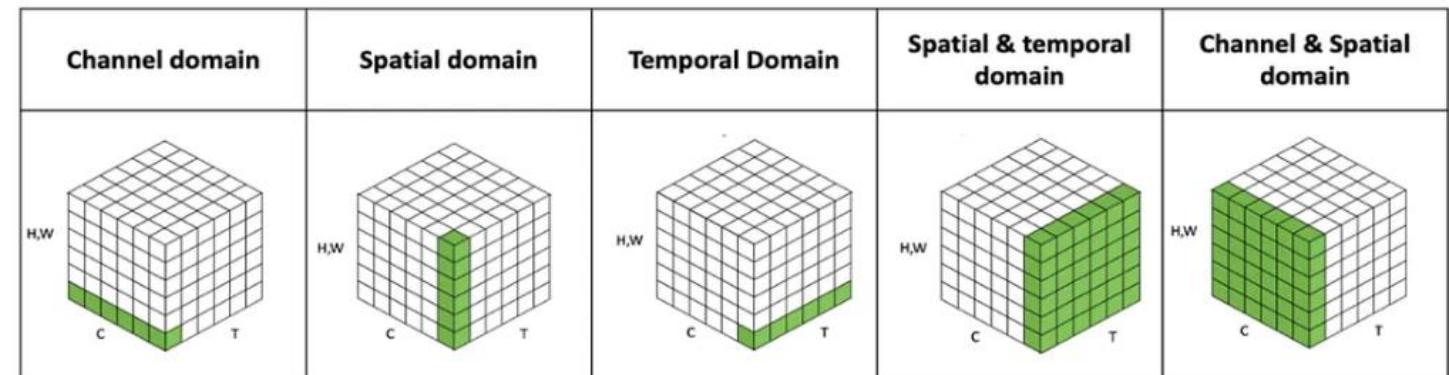
■ Data Domain Categories in Computer Vision Attention

• General Concept

- In computer vision, an **attention mechanism** is a **dynamic weight adjustment function** applied to a feature map **between convolutional layers**.
- Purpose: Guide the next network layer to focus on **more important features** while suppressing less relevant ones.
- Formally: $y = f(g(x), x)$
 - ✓ where $g(x)$ generates an attention map for important regions and $f(g(x), x)$ processes the input feature map x accordingly.

• Four Fundamental Attention Domains

- Channel Attention, Spatial Attention, Temporal Attention, Branch Attention, Hybrid



Dimensions of input tensor: H = height, W = width, C = channel, T = time.

Data domains that different attention mechanisms operate on.

Attention in Computer Vision

■ Data Domain Categories in Computer Vision Attention

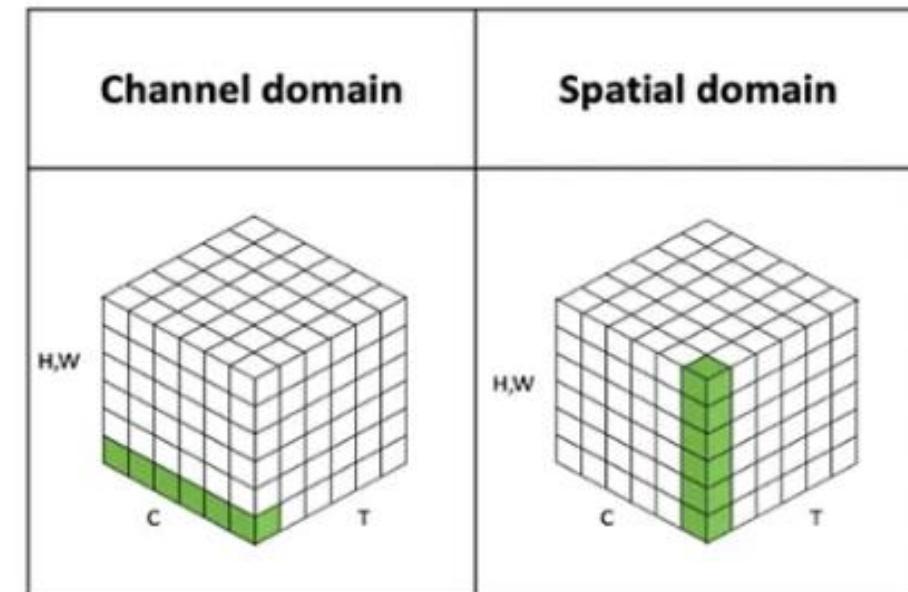
• Four Fundamental Attention Domains

○ 1. Channel Attention – “What to Attend to”

- ✓ Focuses on **feature channels** that are most relevant for the task.
- ✓ Learns channel-wise weights to emphasize informative features and suppress less useful ones.
- ✓ Example Applications: Object classification, fine-grained recognition.

○ 2. Spatial Attention – “Where to Attend to”

- ✓ Highlights important **spatial regions** in the feature map.
- ✓ Often used to localize objects or important regions in an image.
- ✓ Example Applications: Object detection, segmentation.



Dimensions of input tensor: H = height, W = width, C = channel, T = time.

Data domains that different attention mechanisms operate on.

Attention in Computer Vision

■ Data Domain Categories in Computer Vision Attention

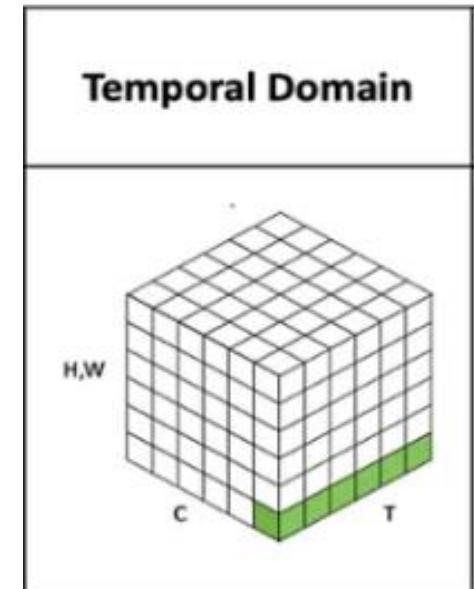
• Four Fundamental Attention Domains

○ 3. Temporal Attention – “When to Attend to”

- ✓ Determines which **time frames** (in video or sequential data) contain critical information.
- ✓ Often combined with spatial attention for video understanding.
- ✓ Example Applications: Action recognition, video captioning.

○ 4. Branch Attention – “Which Path to Attend to”

- ✓ Chooses among **multiple network branches** or kernels adaptively.
- ✓ Enables dynamic routing of information through different processing paths.
- ✓ Example Applications: Multi-scale feature extraction, adaptive convolution.

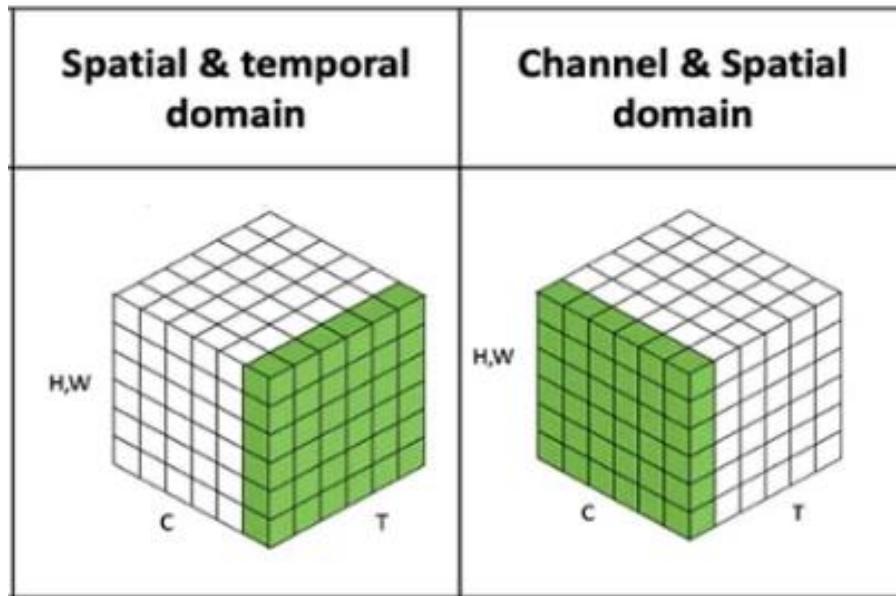


Dimensions of input tensor: H = height, W = width, C = channel, T = time.

Data domains that different attention mechanisms operate on.

Attention in Computer Vision

- Data Domain Categories in Computer Vision Attention
 - Four Fundamental Attention Domains
 - 5. Hybrid Attention Mechanisms



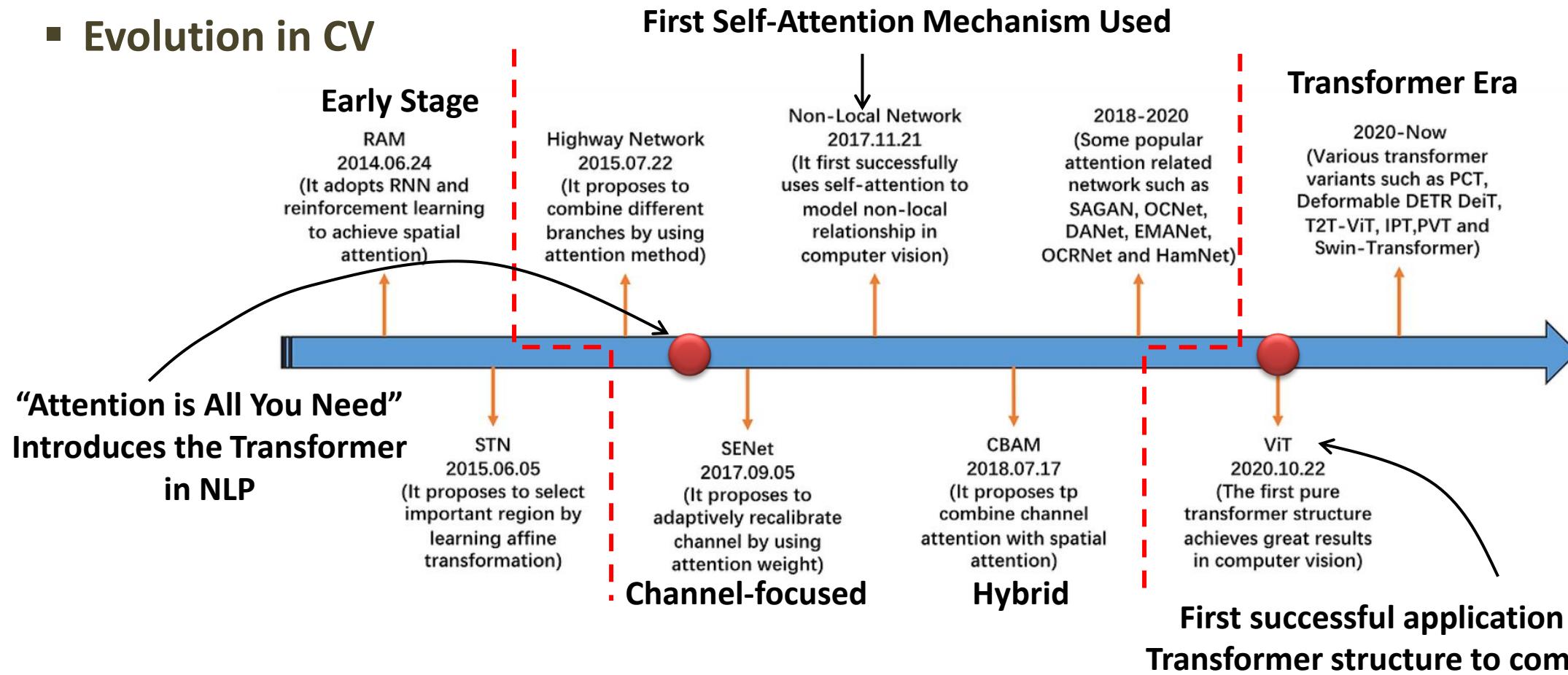
Dimensions of input tensor: H = height, W = width, C = channel, T = time.

Data domains that different attention mechanisms operate on.

- ✓ **Channel + Spatial** → Leverages both **what** and **where** to attend (e.g., combined weighting for channels and pixels).
- ✓ **Spatial + Temporal** → Integrates **where** and **when** to attend (e.g., spatiotemporal attention in video models).

Attention in Computer Vision

■ Evolution in CV



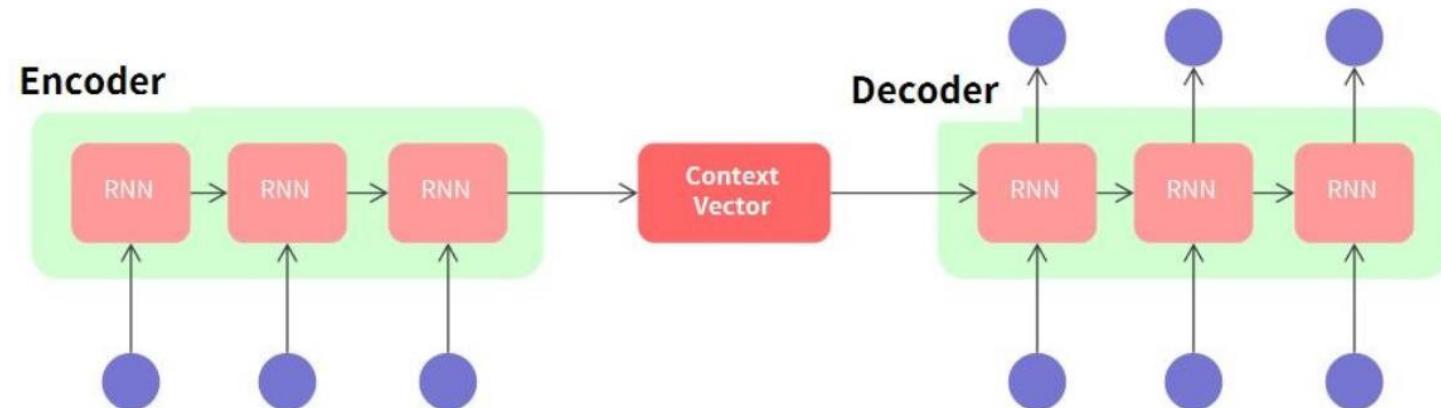
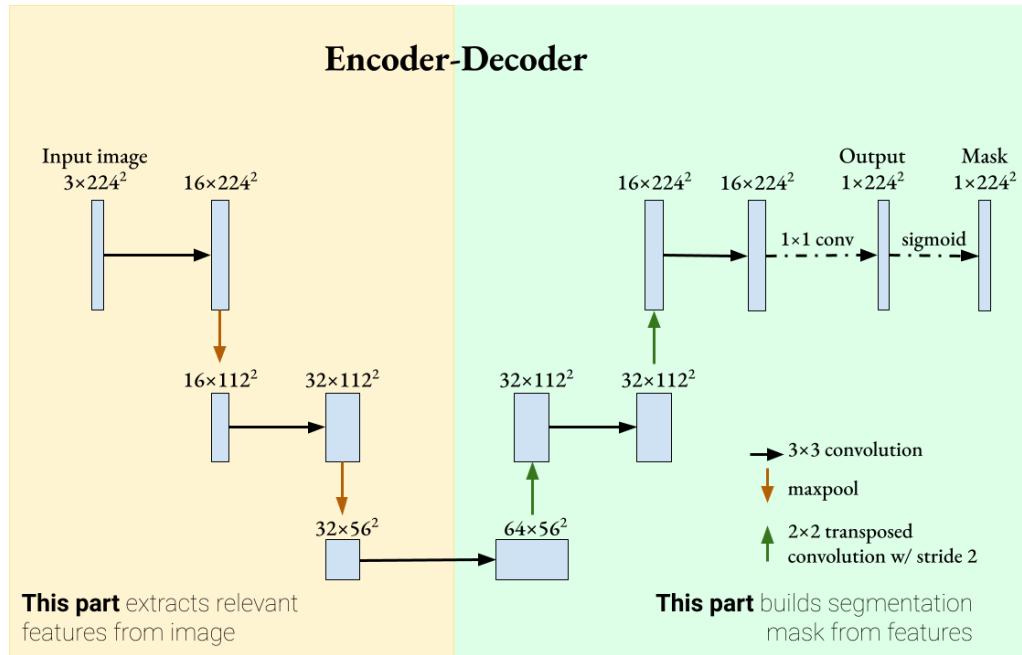
- We will explore the **Transformer architecture** in detail and study various **Transformer-based CV models** such as ViT, Swin Transformer, and PVT.

Why a Deeper Look at Encoder–Decoder?

- We've already met **U-Net** (CV) and **Seq2Seq** (NLP).
- Before we study **Transformers**, let's solidify the **Encoder–Decoder pattern**: what it is, how information flows, and where it breaks.
- Learning objectives
 - Understand the data pipeline (tokens/patches → embeddings → latent/context)
 - See how **decoders** generate outputs (sequence vs. image)
 - Identify bottlenecks that motivate **attention** and **Transformers**

Why a Deeper Look at Encoder–Decoder?

- We've already met **U-Net (CV)** and **Seq2Seq (NLP)**.



- Before we study **Transformers**, let's solidify the **Encoder–Decoder pattern**: what it is, how information flows, and where it breaks.

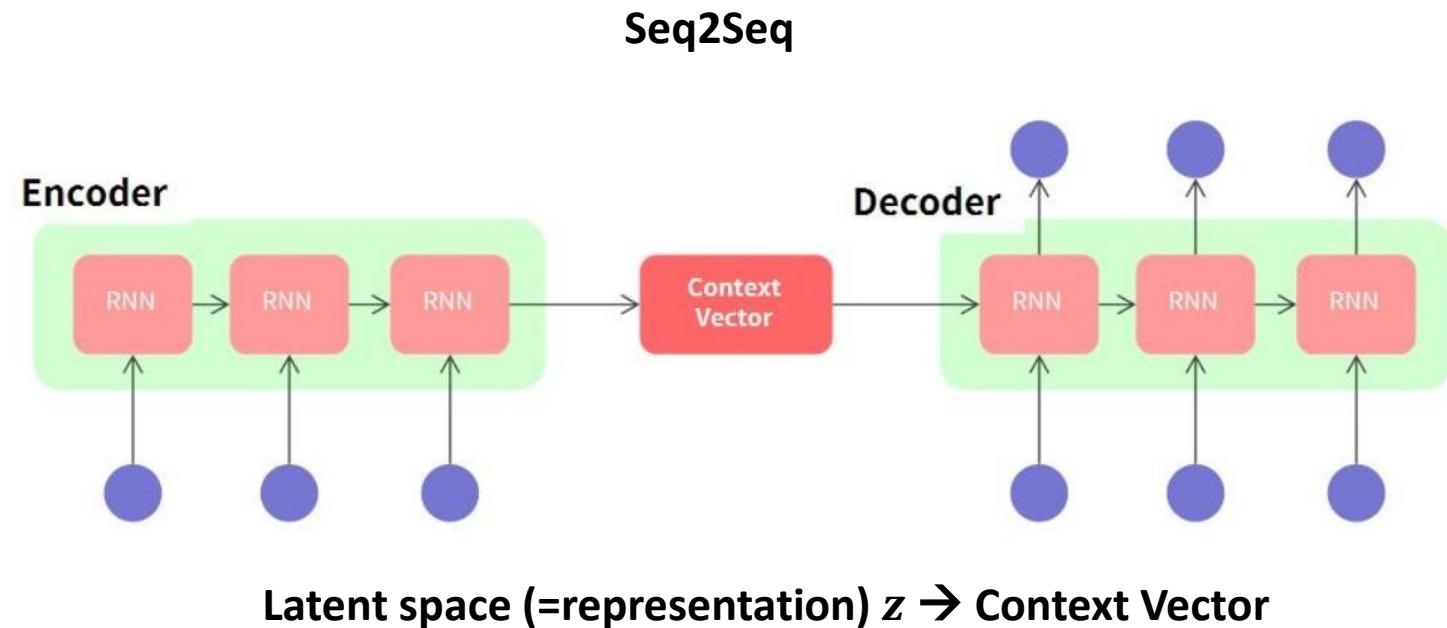
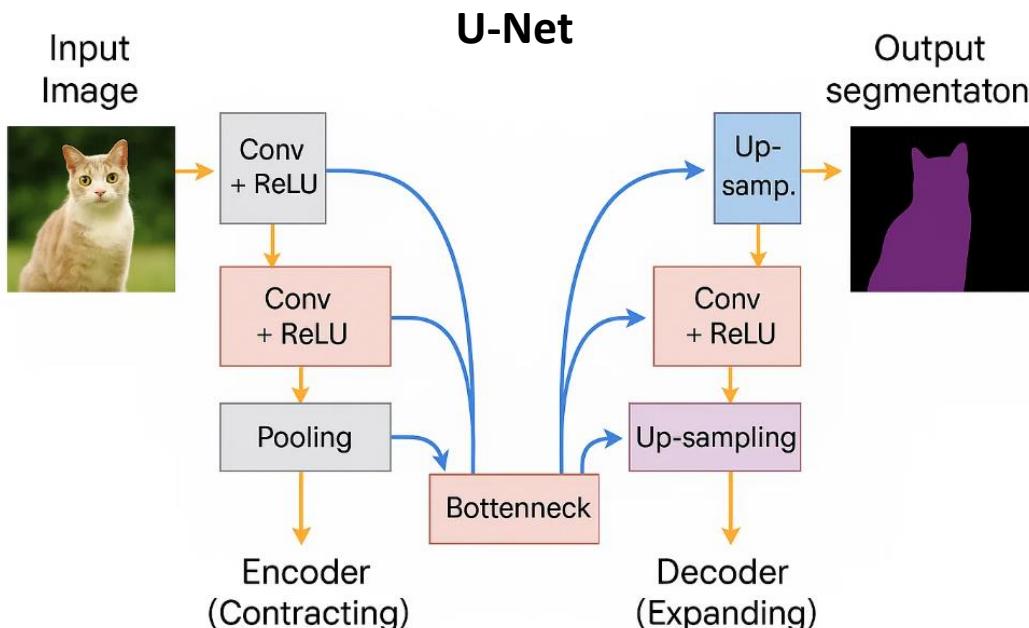
- Learning objectives**

- Understand the data pipeline (tokens/patches → embeddings → latent/context)
- See how **decoders** generate outputs (sequence vs. image)
- Identify bottlenecks that motivate **attention** and **Transformers**

Why a Deeper Look at Encoder–Decoder?

■ The Encoder–Decoder Mental Model

- Mapping $x \rightarrow z \rightarrow y$



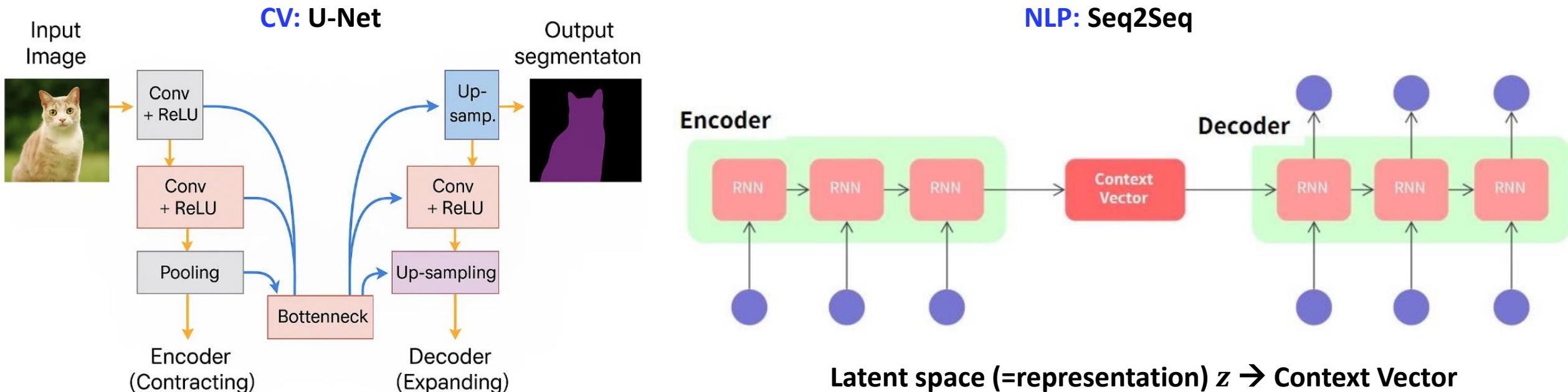
Latent space (=representation) $z \rightarrow$ Bottleneck

- **Encoder:** compresses input into a **latent representation z** (a vector, grid, or pyramid).
- **Decoder:** expands z into an output with task-specific structure y (sequence, mask, image).

Why a Deeper Look at Encoder–Decoder?

■ The Encoder–Decoder Mental Model

- Same idea, different instantiations



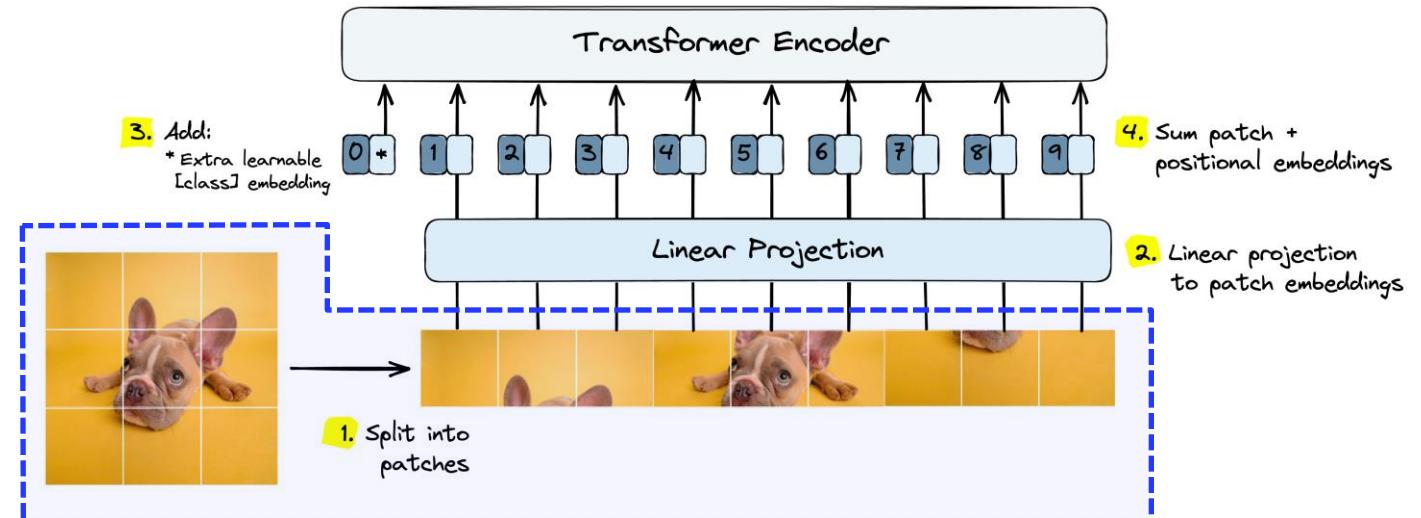
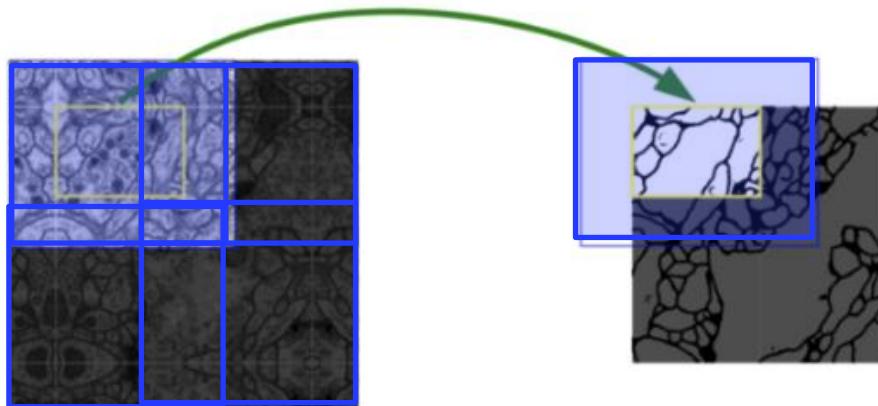
Latent space (=representation) $z \rightarrow$ Bottleneck

- **NLP:** RNN/GRU/LSTM encoder \rightarrow vector \rightarrow RNN decoder (Seq2Seq)
- **CV:** CNN encoder (downsampling) \rightarrow feature pyramid \rightarrow CNN decoder (upsampling)
- Optional: **skip connections** (e.g., U-Net) to restore spatial detail.=

Why a Deeper Look at Encoder–Decoder?

- CV: Patchification / NLP: Tokenization (Input → Units)
 - Vision (U-Net & friends)

Patch : image recognition unit



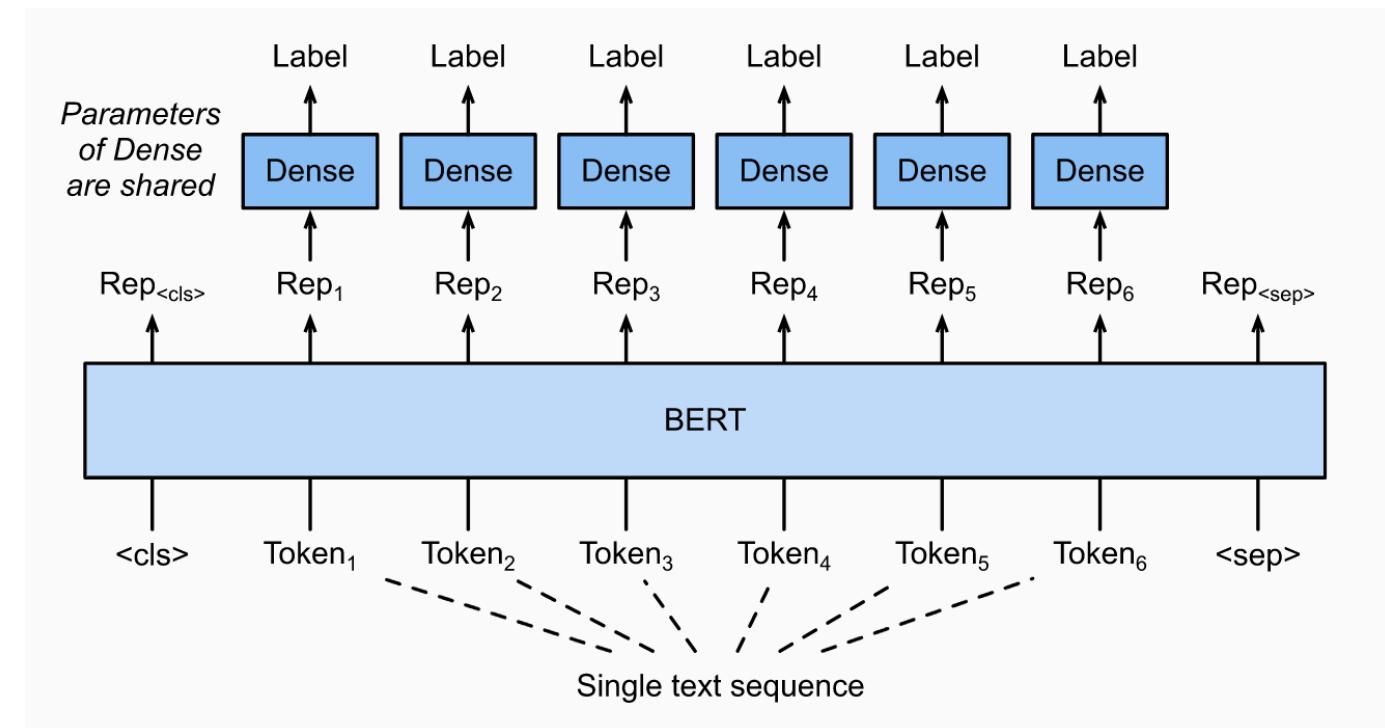
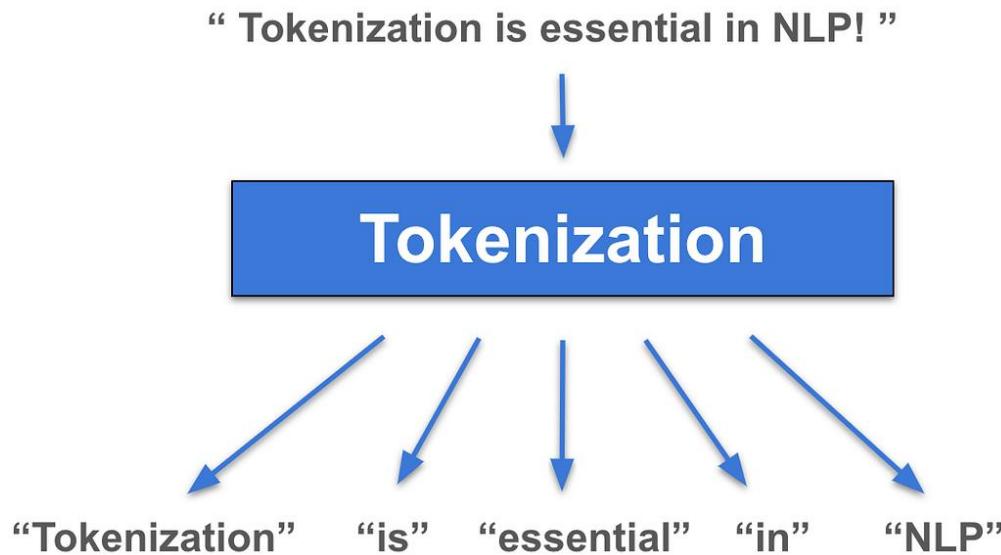
U-Net: Overlap-tile (i.e., Overlapped Multiple Patches)

- Convolutions split the image into a **feature grid** (implicit tokens).
- (Preview for ViT) Images can be **patchified** into fixed-size tokens (e.g., $16 \times 16 \times 16$).

Why a Deeper Look at Encoder–Decoder?

■ CV: Patchification / NLP: Tokenization (Input → Units)

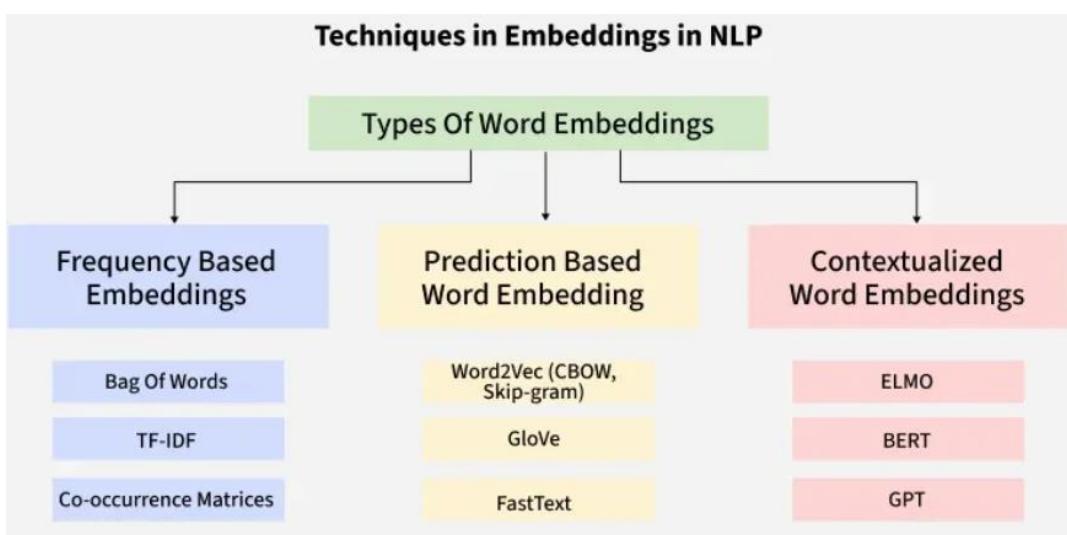
- Text (Seq2Seq)



- Tokenize into words/ subwords / characters.
- Add special tokens: <START>, <END>, <PAD>.
- **Outcome:** a sequence or grid of units (i.e., patches or tokens) that the encoder can embed.

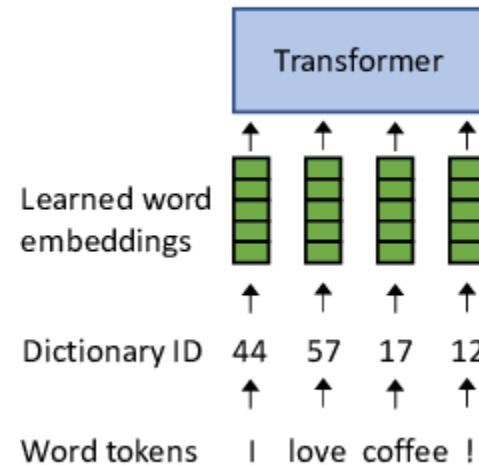
Why a Deeper Look at Encoder–Decoder?

■ Embedding (Units → Vectors)

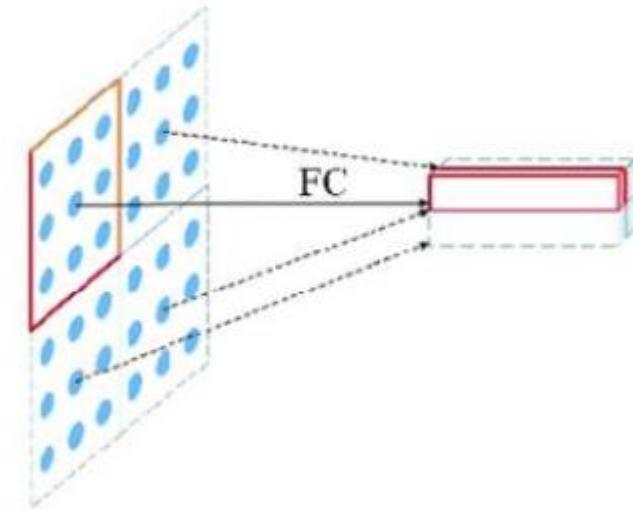


Word Embedding in NLP

Domain: NLP



Patch Embedding in NLP



- **Text:** map tokens to dense vectors (Word2Vec/GloVe/FastText or learned embeddings).
- **Vision:** first conv layers act as **learned embedding** of local patches; in ViT, a linear “patch embedding” projects flattened patches.
- **Why embeddings matter:**
 - Move from symbolic space to a **continuous, learnable geometry** where proximity encodes meaning/appearance.

Why a Deeper Look at Encoder–Decoder?

■ General Encoder-Decoder Architecture

- **1. The Encoder (Compress & Organize Information)**

- Text (RNN encoder)
 - ✓ Processes embeddings left→right; final hidden state(s) summarize input.
- Vision (CNN encoder)
 - ✓ Stacks conv + pooling/strides → **lower H×W, higher channels**; captures multi-scale semantics.
- Output forms
 - ✓ **(1) Single vector** (classic Seq2Seq), **(2) Sequence of states** (one per token), and **(3) Feature maps/pyramids** (for images)

- **2. Context Vector (The Bottleneck)**

- **Classic Seq2Seq:** a **fixed-length vector c** (e.g., last hidden state) is the only conduit to the decoder.
 - ✓ Pros: simple; Cons: **information bottleneck**, especially for long inputs.
- **CV contrast:** U-Net mitigates a similar bottleneck with **skip connections** (high-res features passed directly to the decoder).
- This sets up the need for **attention** (dynamic, content-aware access to encoder states).

Why a Deeper Look at Encoder–Decoder?

■ General Encoder-Decoder Architecture

- **3. The Decoder (Generate the Output)**

- **Text (sequence decoder)**

- ✓ Initialize with context; at time t , use previous token + hidden state → predict next token (softmax).
 - ✓ Training tricks: **teacher forcing, scheduled sampling, beam search** at inference.

- **Vision (image/segmentation decoder)**

- ✓ Upsampling (transpose conv / interpolation + conv) to reconstruct spatial detail.
 - ✓ May fuse **skip connections** for sharp boundaries (U-Net).

- **Putting It Together (Two Pipelines)**

- **NLP pipeline:** Tokens → Embeddings → RNN Encoder → (Context or Attention) → RNN Decoder → Sequence
 - **CV pipeline:** Image → Conv Encoder → Feature Pyramid (+ Skips) → Conv Decoder → Mask/Image
 - **Shared idea: encode structure → decode structure; differences are the unit type (tokens vs. pixels/features) and decoder objective.**

Why a Deeper Look at Encoder–Decoder?

- Where Things Break (Motivation for Attention/Transformer)
 - 1. Fixed-length **context vectors** struggle with long or complex inputs.
 - 2. Sequential RNN decoders limit **parallelism** and **long-range reasoning**.
 - 3. In CV, plain upsampling can miss **global context** without extra links.
 - 4. **Attention** (cross-/self-) fixes the access problem; **Transformers** replace recurrence with **parallel attention blocks**.
- Quick Recap & Transition
 - We now understand **Encoder–Decoder** across NLP & CV
 - Tokens/patches → embeddings → encoder states → (context/attention) → decoder.
 - Next up
 - **Transformers** — same high-level blueprint, but replace recurrence with **multi-head self-attention**, add **positional encodings**, and scale with **parallel computation**.