Computer vision

Final project

IMAGE CAPTIONING

522K0031 – Pham Thai An Binh

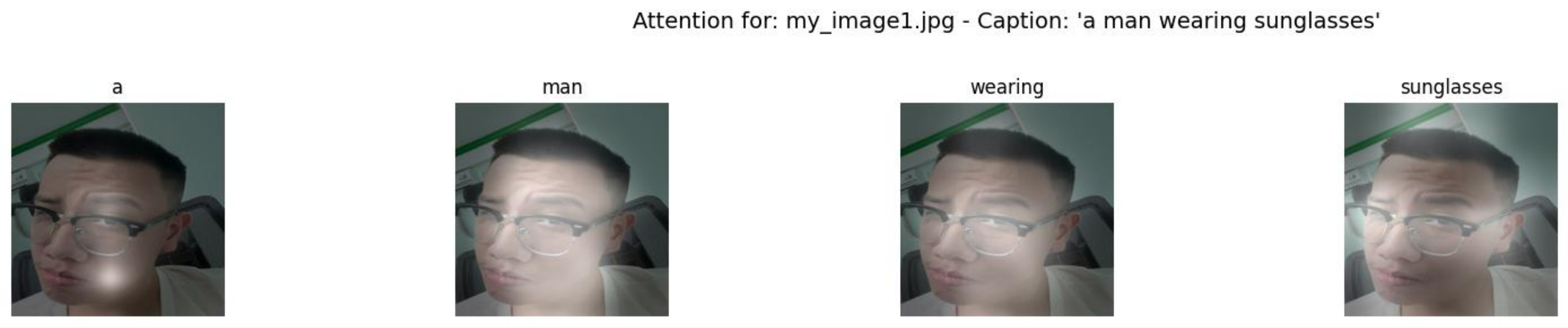
522K0038 – Nguyen Nhat Khoa

What is Image Captioning?

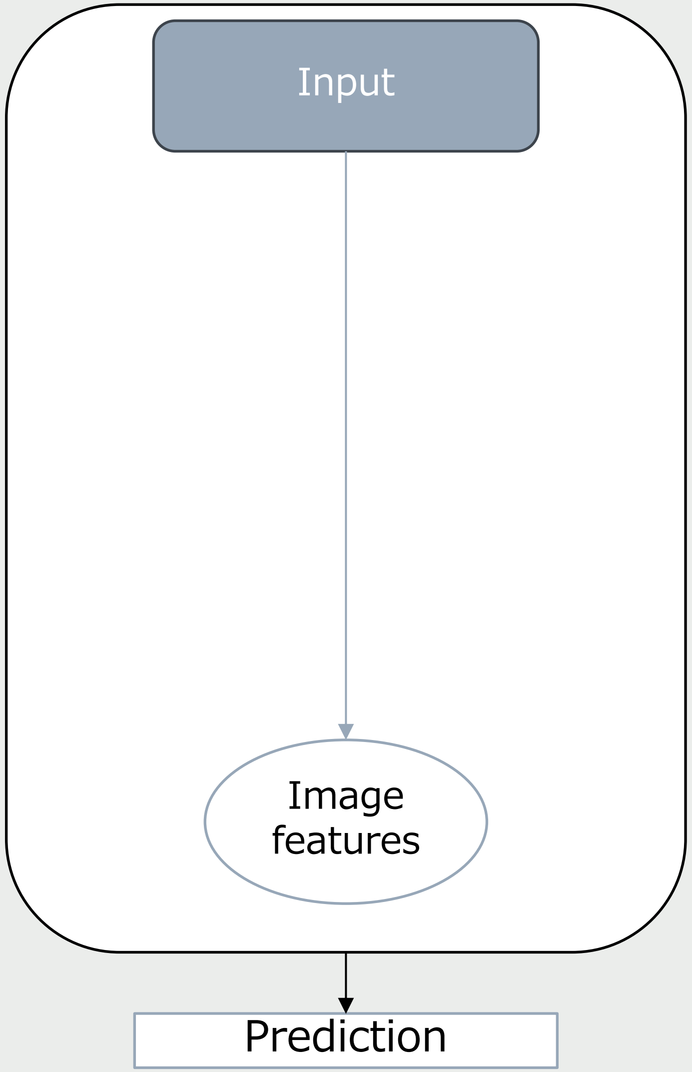
Generating a natural language description for a given image

Accessibility, Image Retrieval, Human and AI Interaction, Robotics

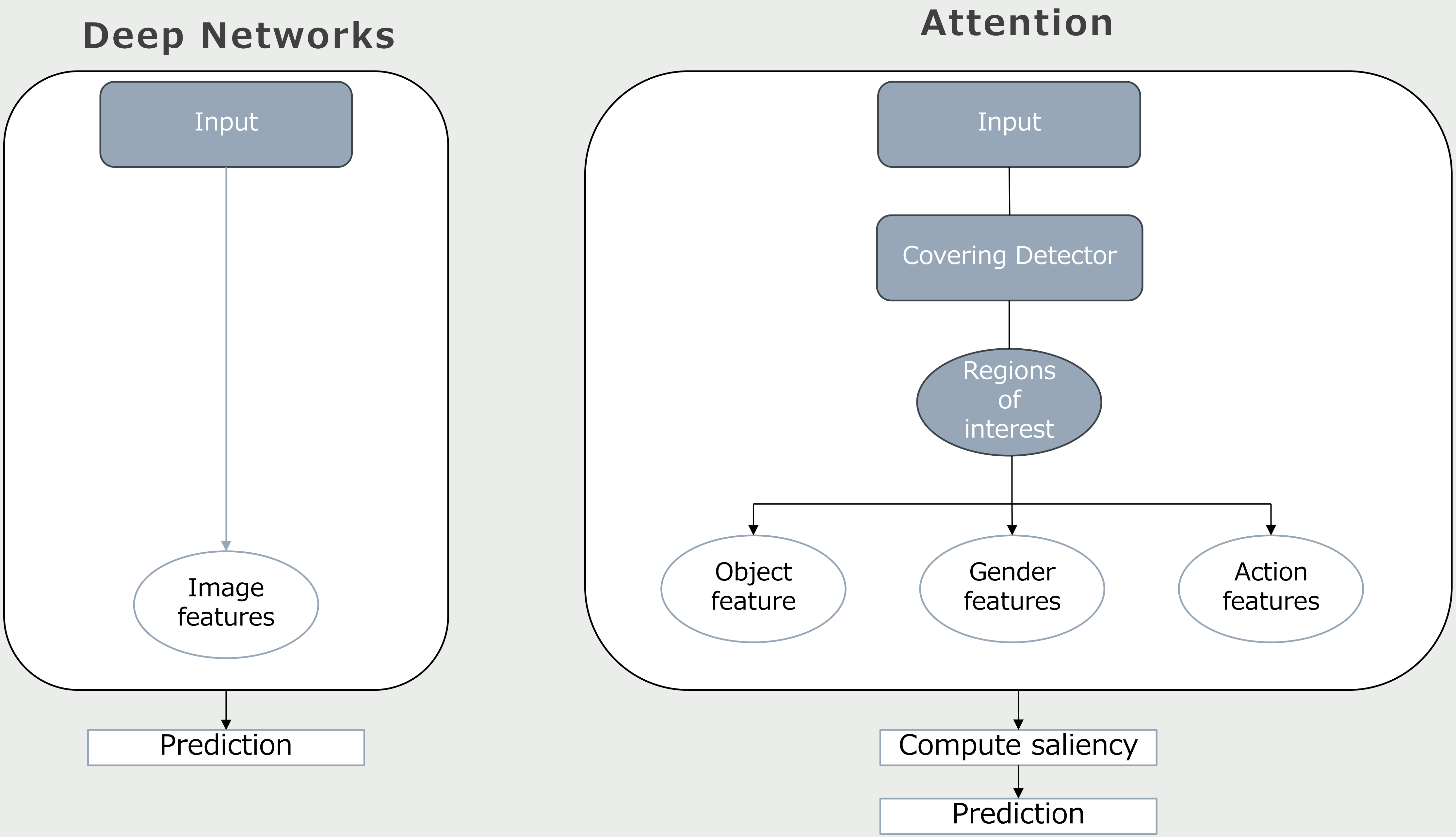
Translating visual scenes into coherent sentences



Deep Networks



Limitations of Earlier Approaches


Deep Networks
Input
Image features
Prediction
Attention
Input
Covering Detector
Regions of interest
Object feature
Gender features
Action features
Compute saliency
Prediction

The Core idea of “Show, Attend and tell”

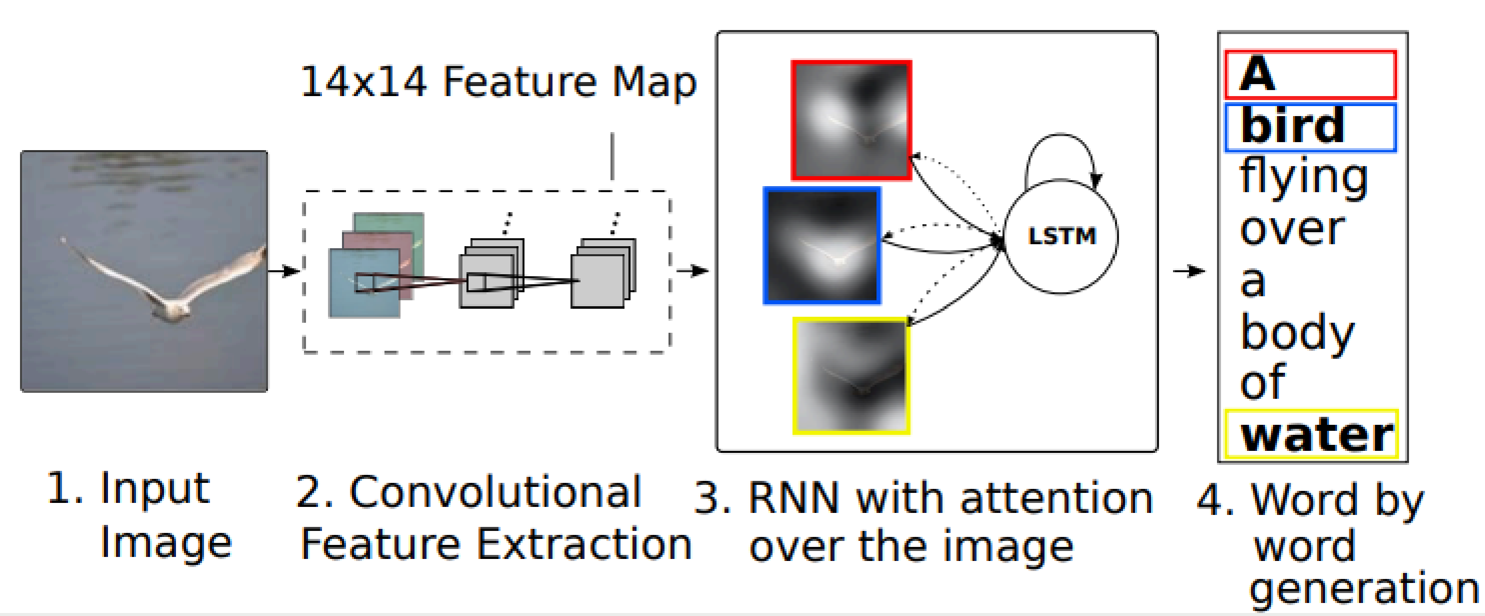
Main Contribution of paper

Introduced **attention-based mechanism**

Model dynamically focuses on important regions of an image.

How the Attention Mechanism Works

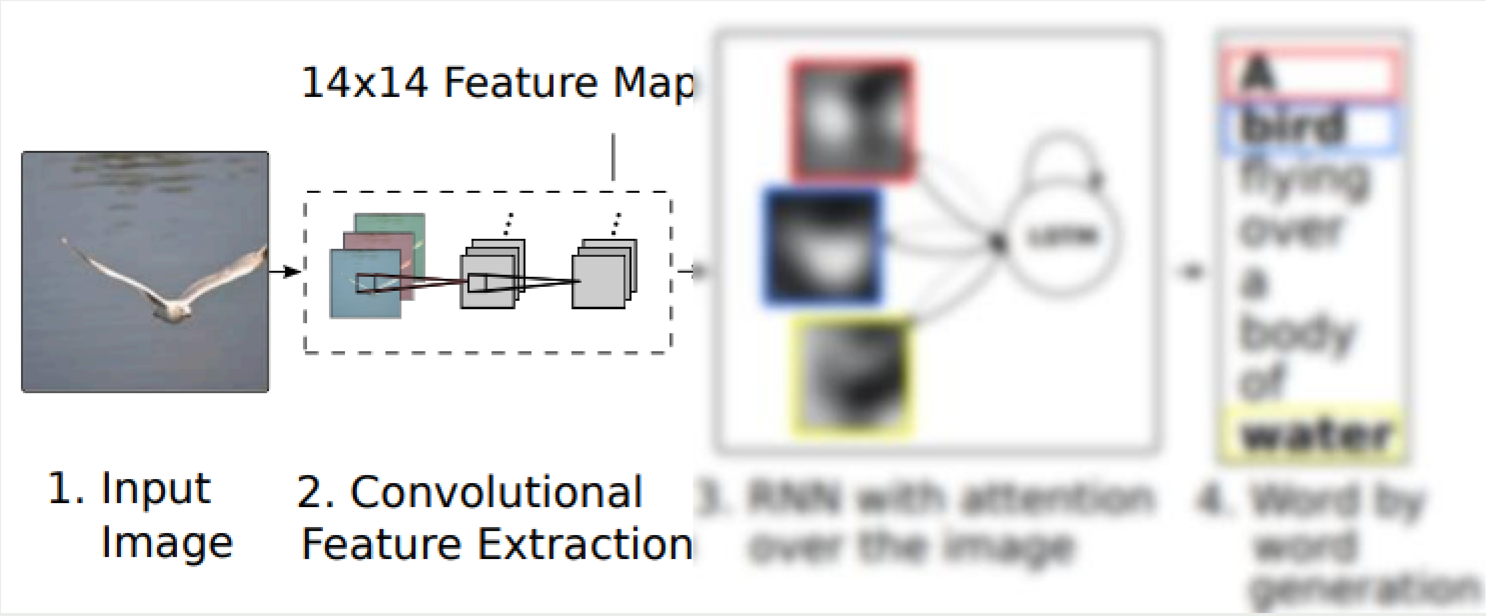
Objective: To determine the most relevant image regions for generating each word in the caption.



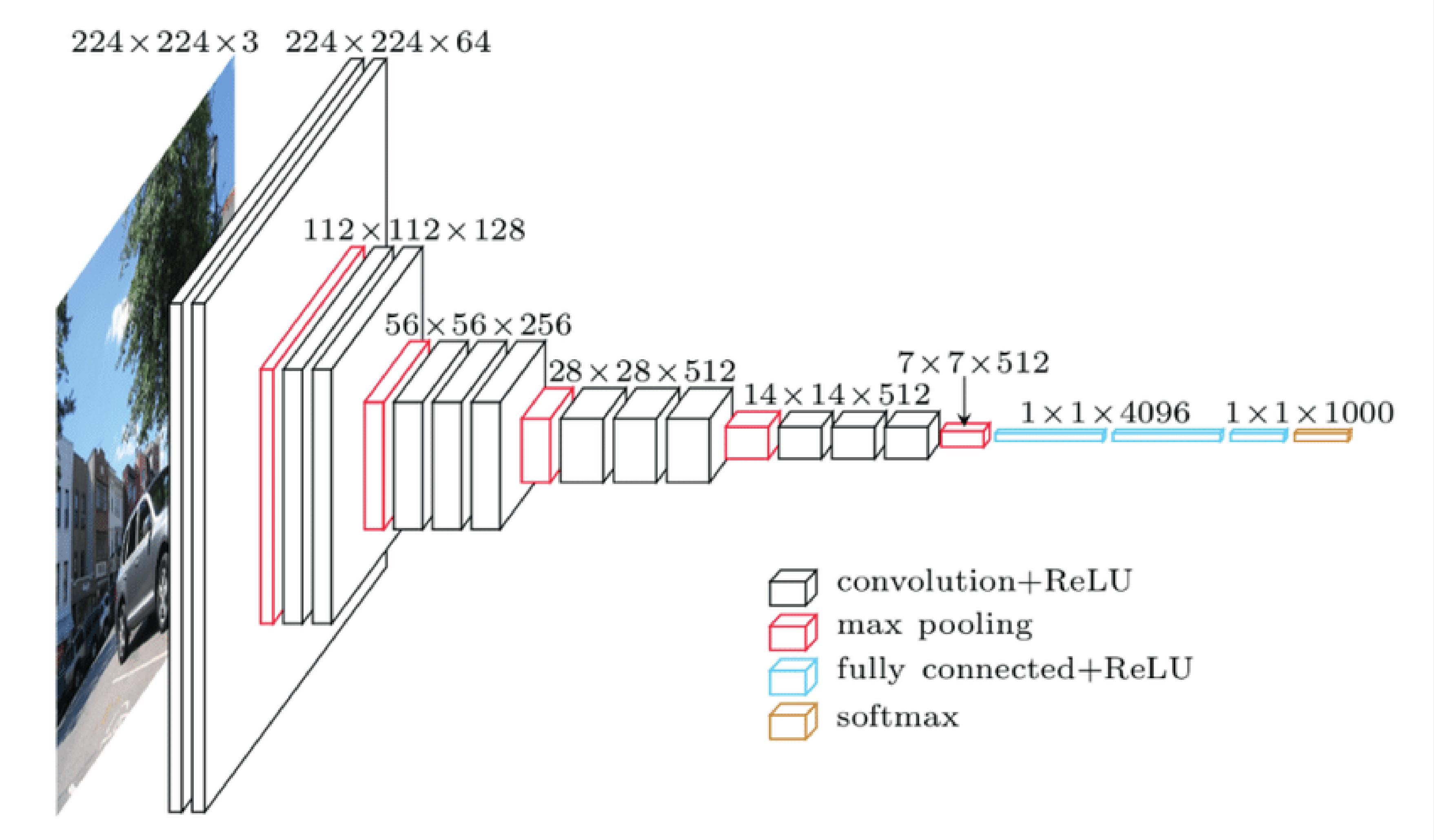
How the Attention Mechanism Works

Objective: To determine the most relevant image regions for generating each word in the caption.

1. An image encoder (CNN) processes the input image. It outputs a grid of feature vectors

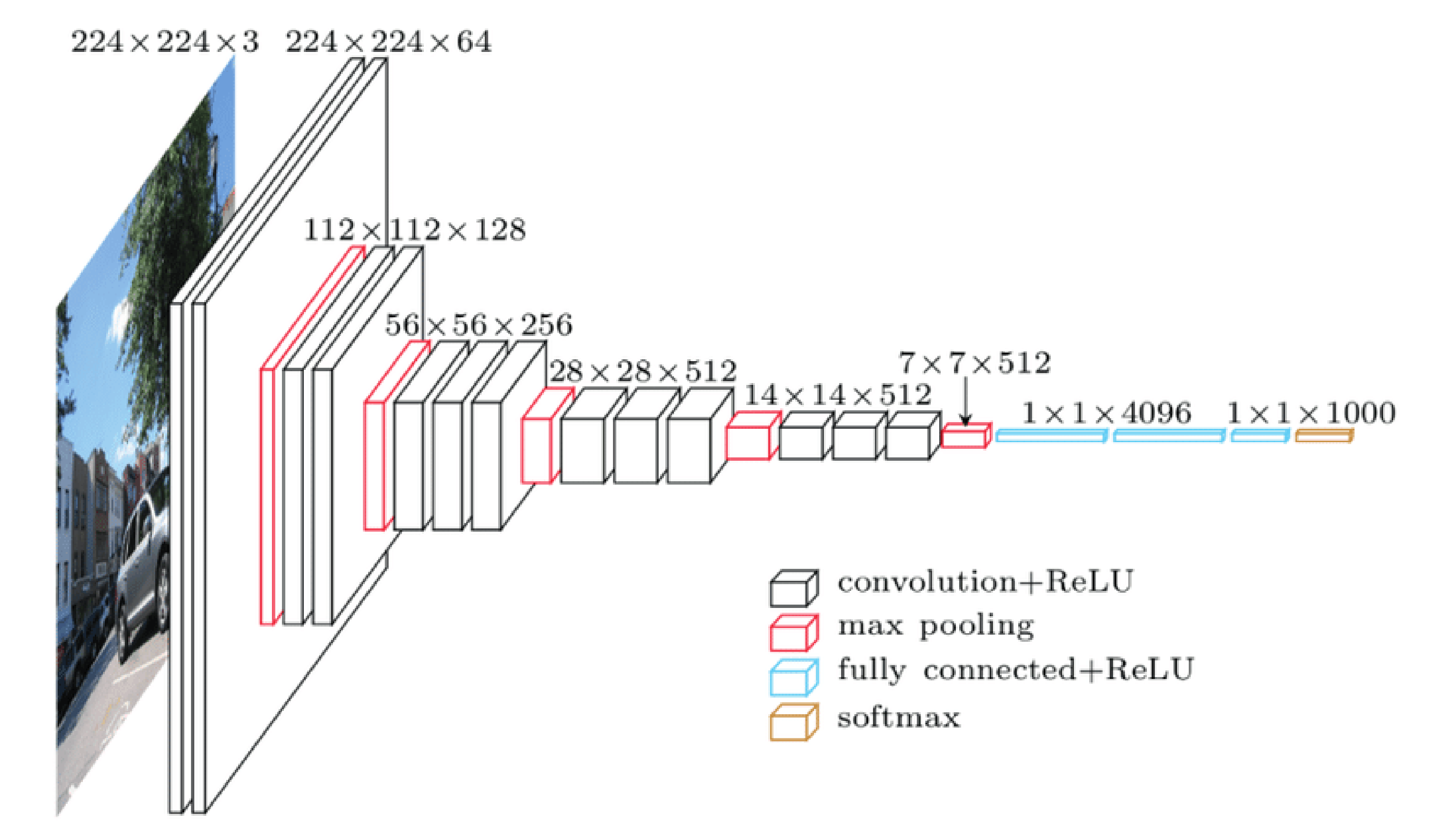


CNN Encoders(VGGNet)



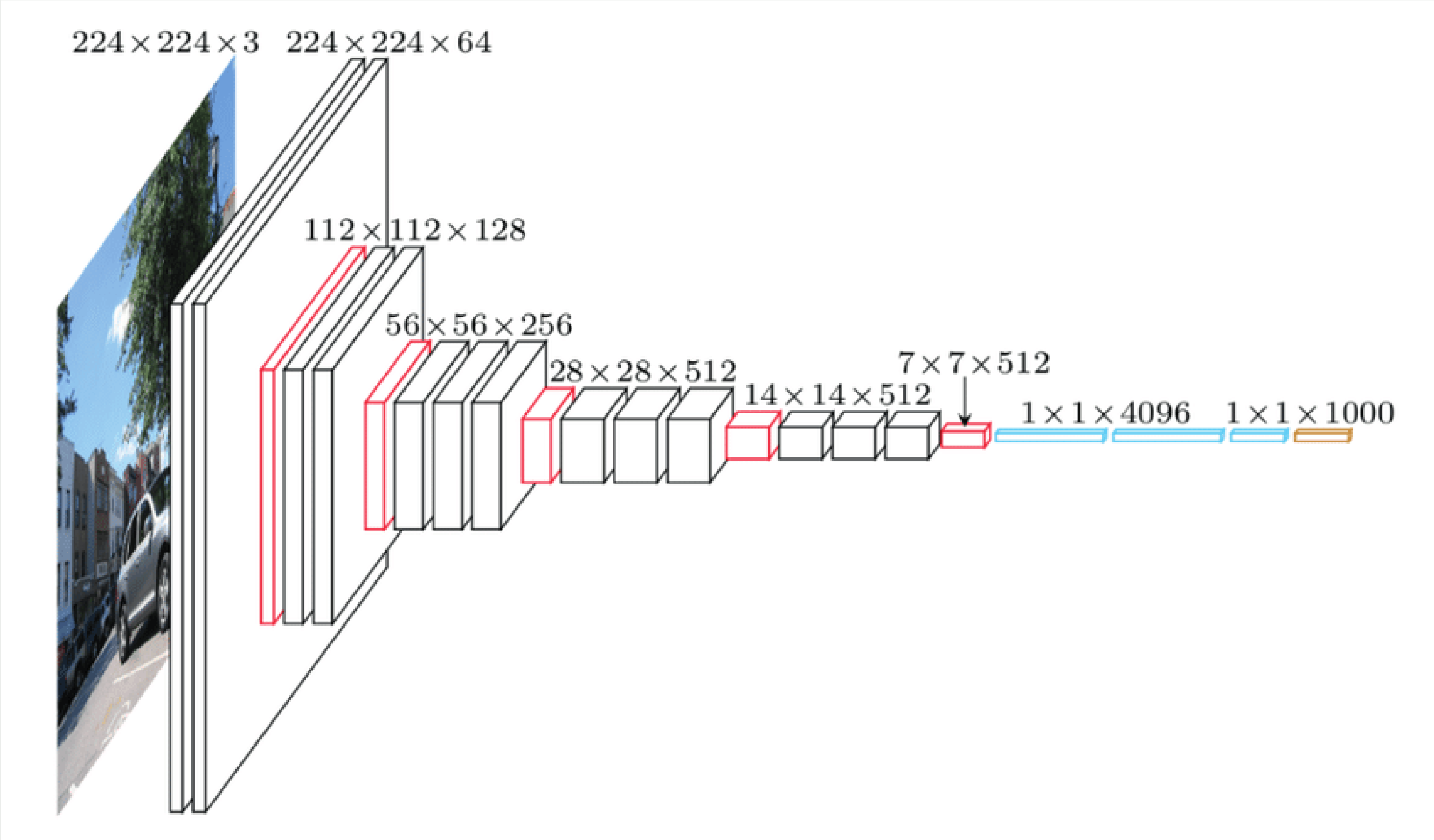
CNN Encoders(VGGNet)

Purpose: The CNN's role is to transform the input image into a rich, spatially organized set of features that the attention mechanism can later use.



CNN Encoders(VGGNet)

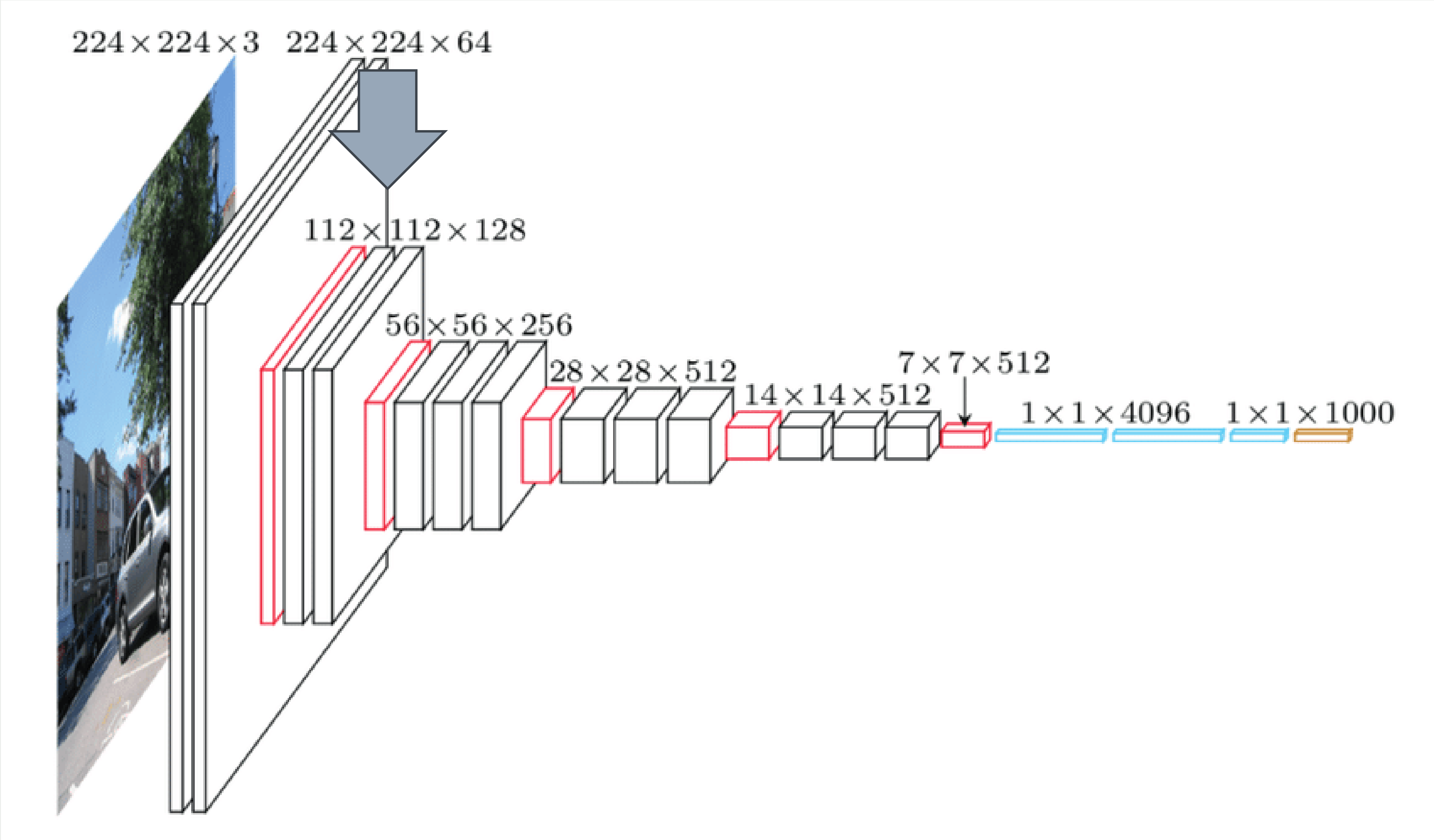
VGGNet processes an input image (ex: 224x224x3) through a series of convolutional layers and max-pooling layers.



CNN Encoders(VGGNet)

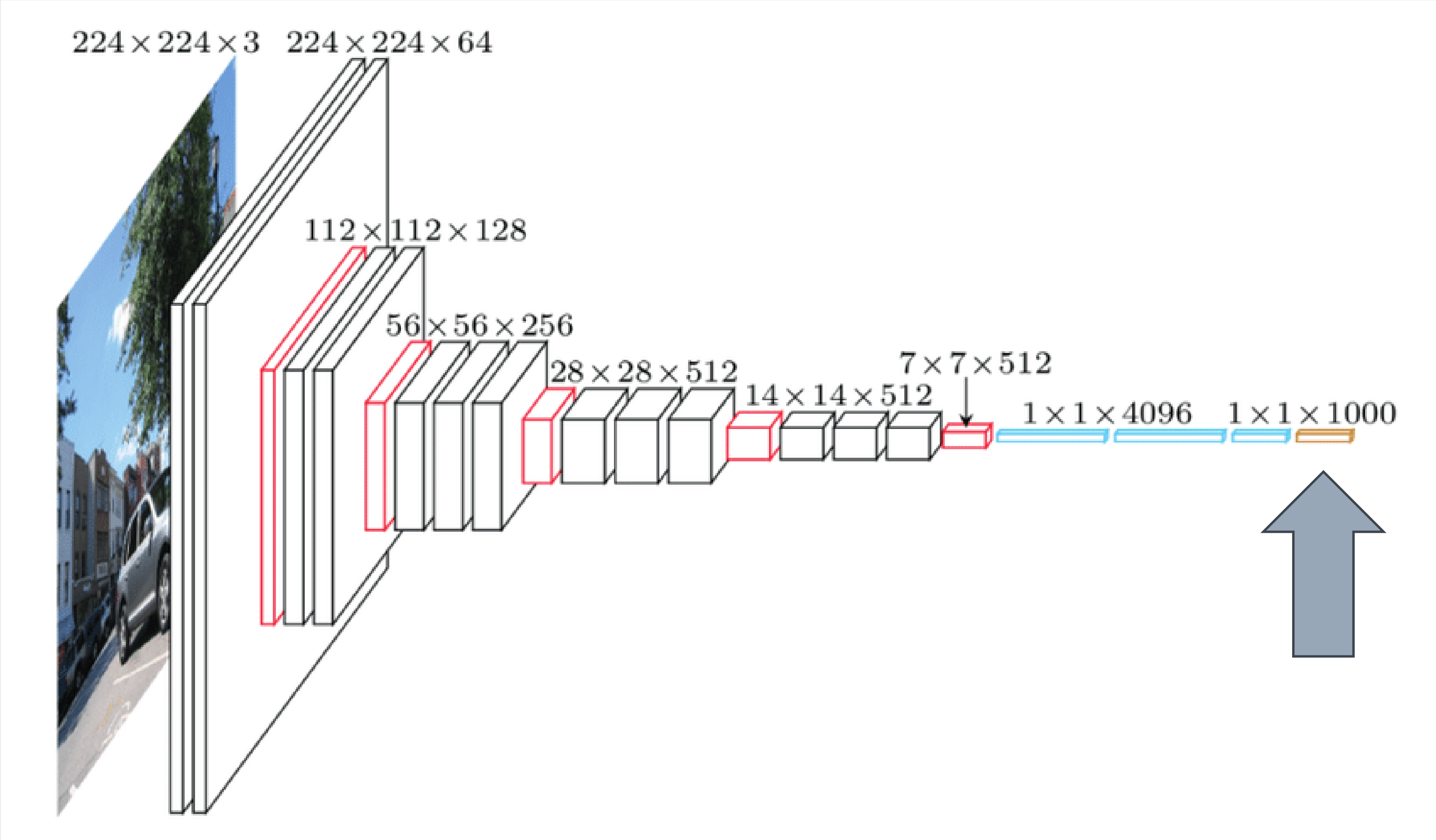
VGGNet processes an input image (ex: 224x224x3) through a series of convolutional layers and max-pooling layers.

Convolutional layers extract features, and max-pooling layers reduce the spatial dimensions



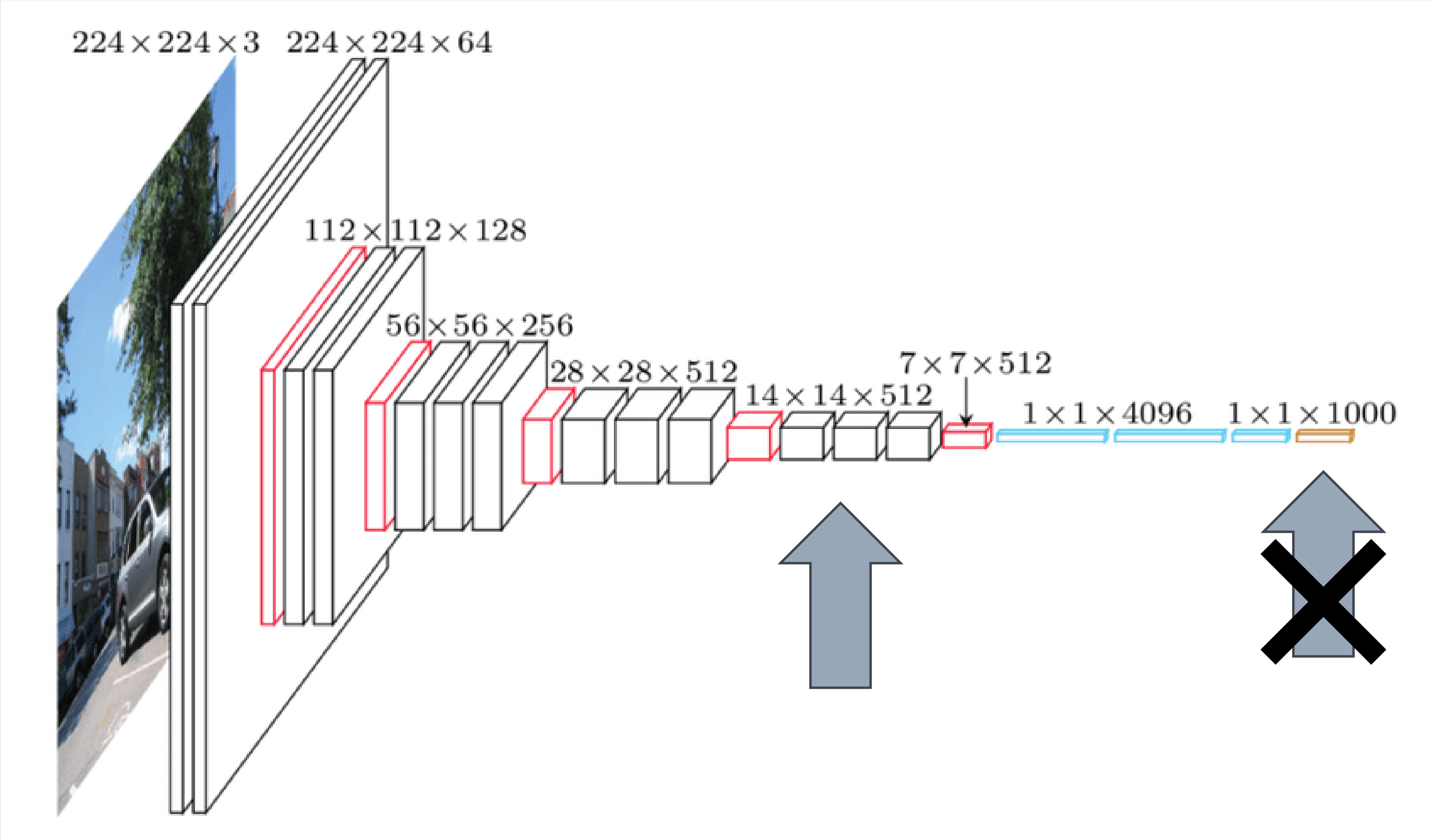
CNN Encoders

Key for Attention: Instead of using the final classification layers , the "Show, Attend, and Tell" paper extracts features from an earlier convolutional layer.

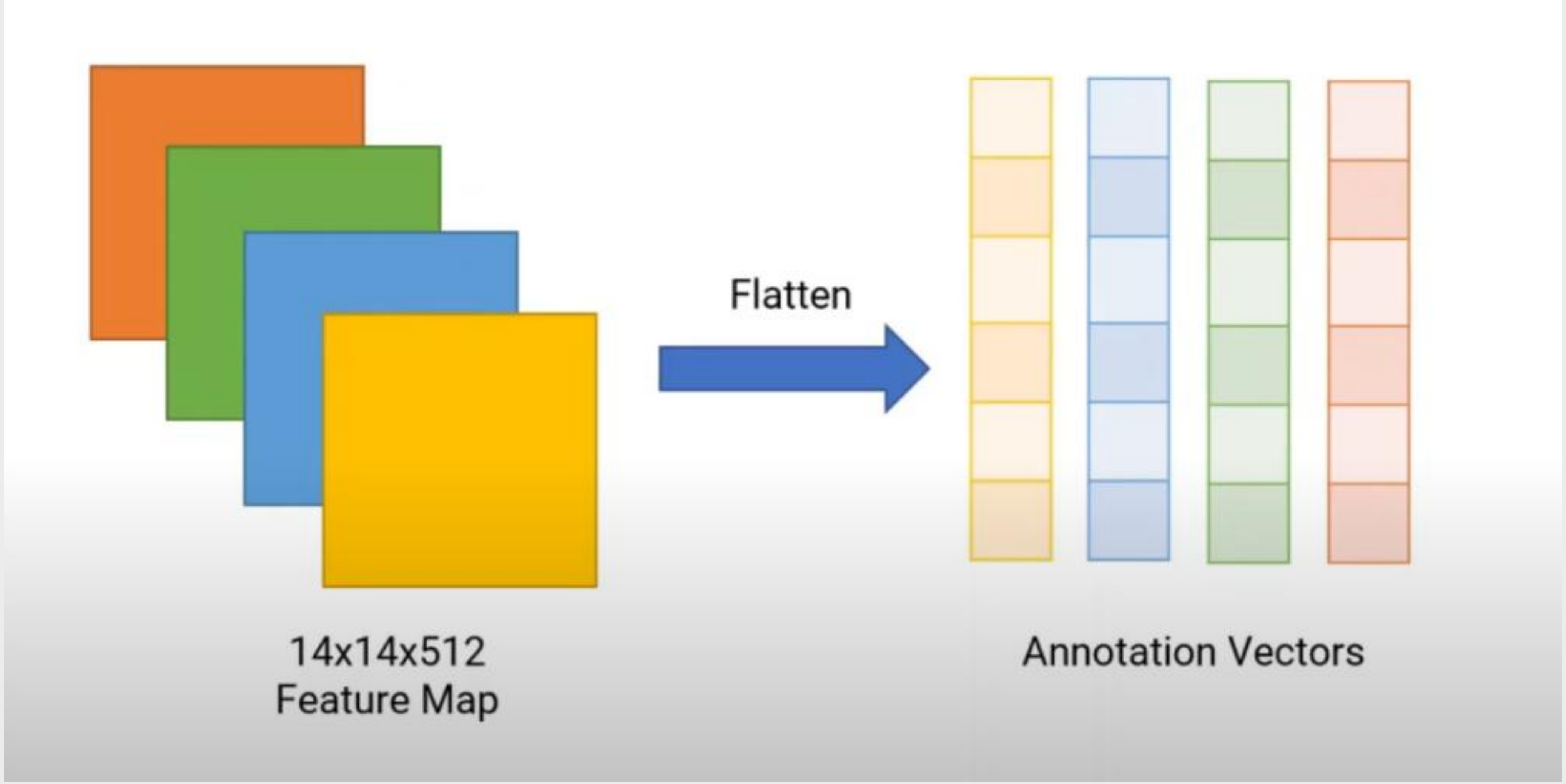


CNN Encoders

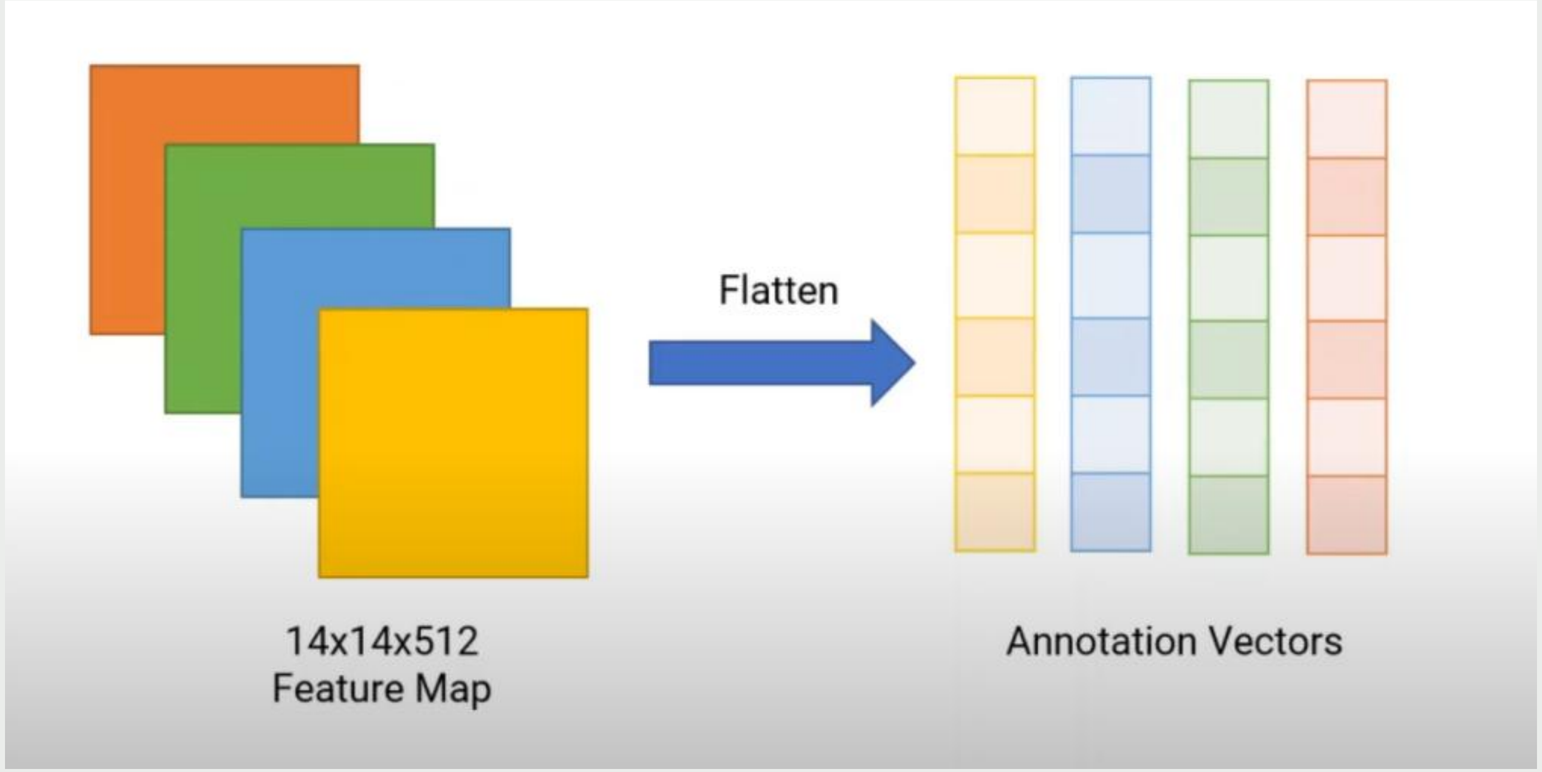
Key for Attention: Instead of using the final classification layers , the "Show, Attend, and Tell" paper extracts features from an earlier convolutional layer.



From Feature Map to Annotation Vectors

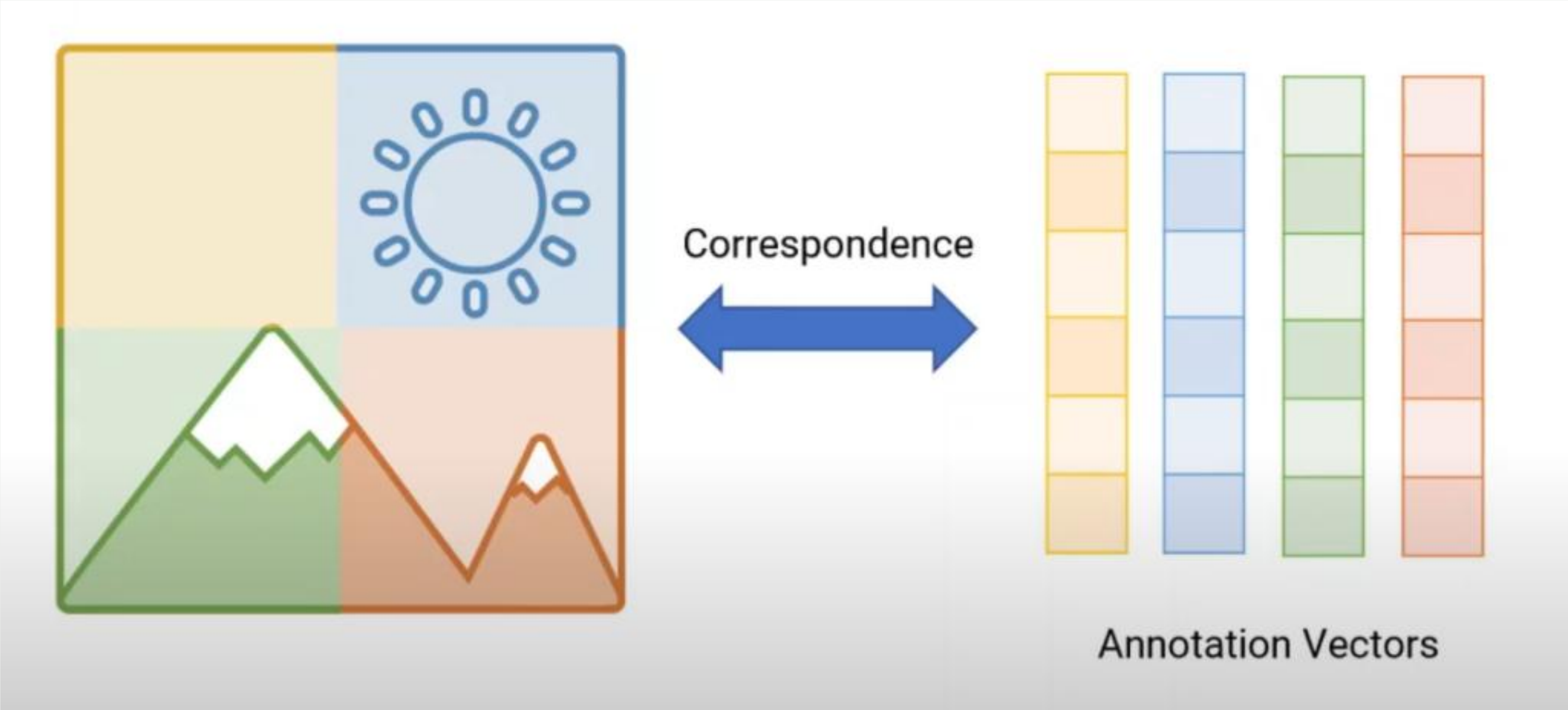


From Feature Map to Annotation Vectors

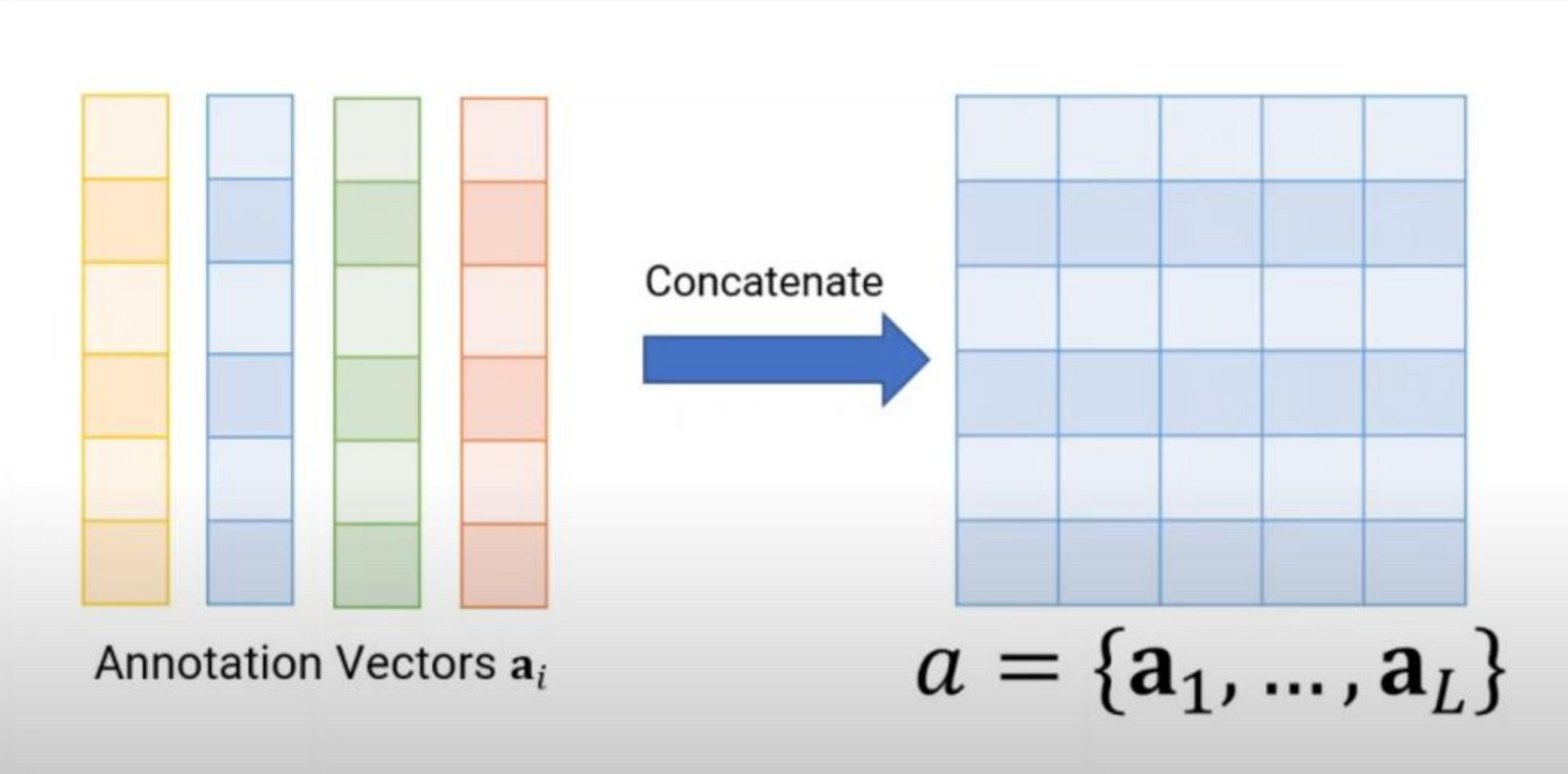


***Flatten:** transforming a matrix into a single, long list or one-dimensional array

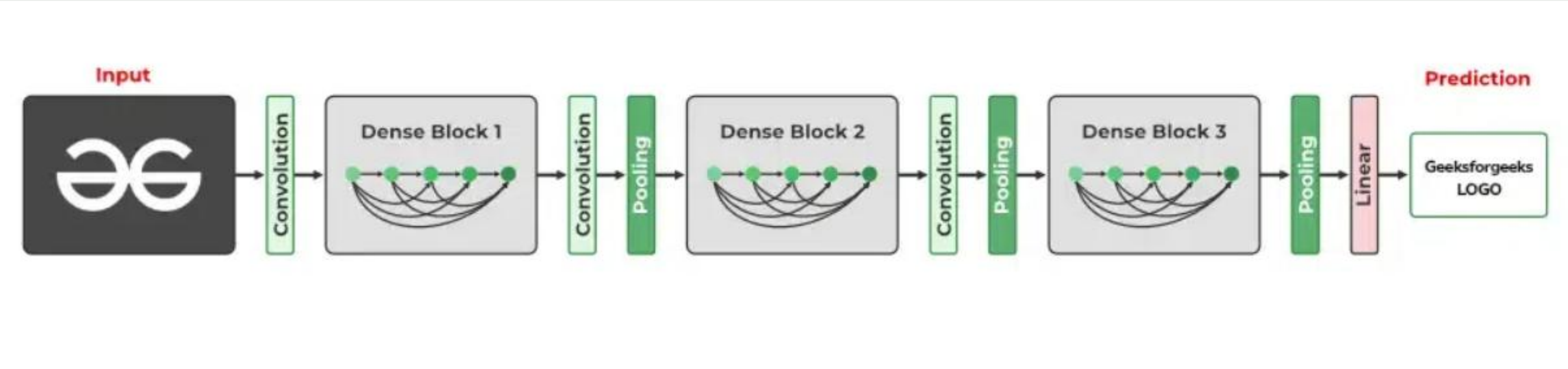
From Feature Map to Annotation Vectors



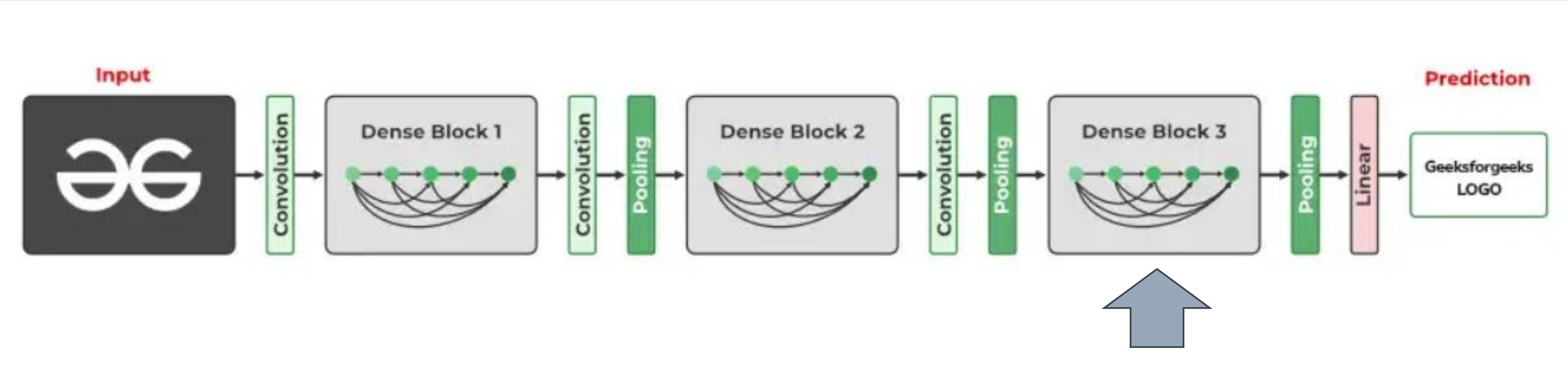
From Feature Map to Annotation Vectors



CNN Encoders(DenseNet-205)



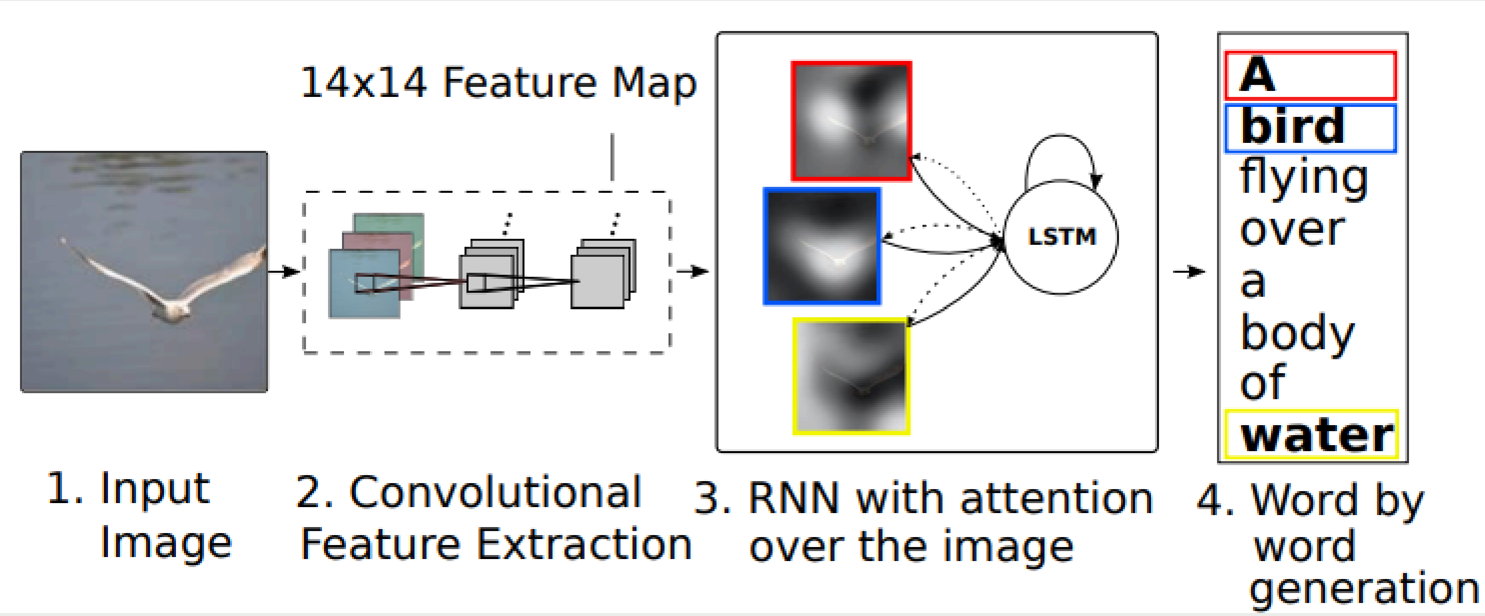
CNN Encoders(DenseNet-205)



How the Attention Mechanism Works

Objective: To determine the most relevant image regions for generating each word in the caption.

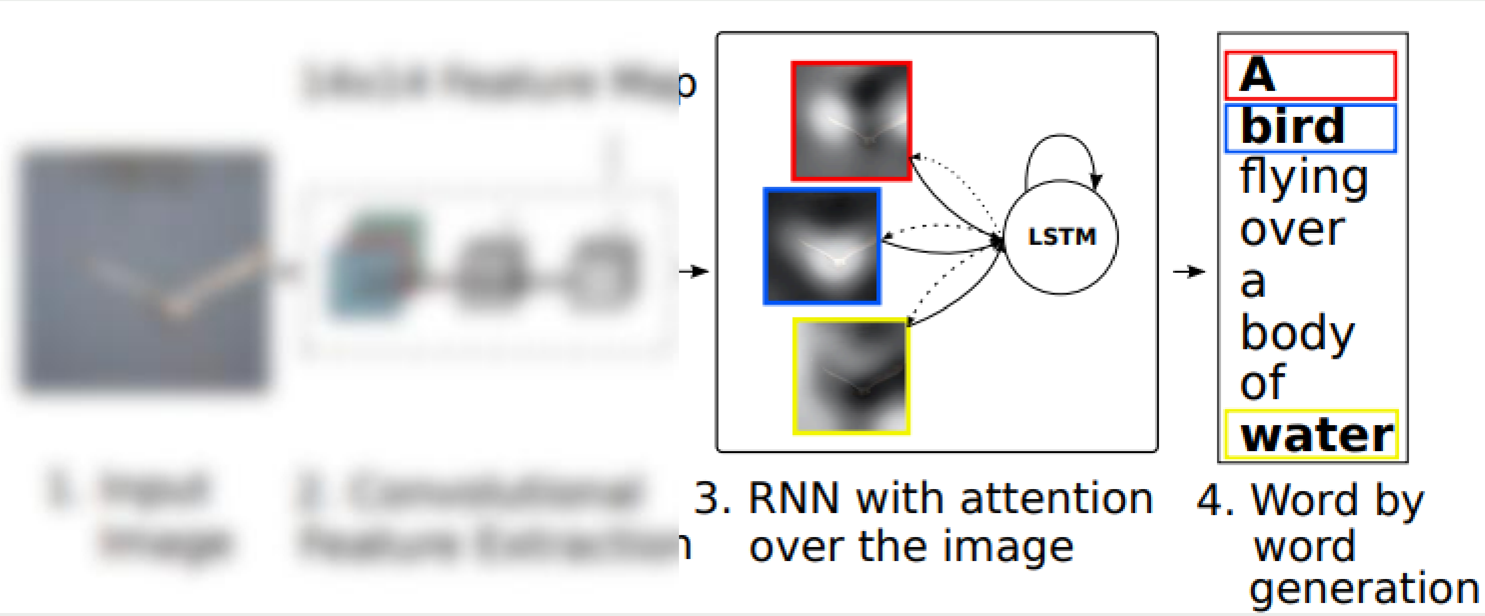
1. An image encoder (CNN) processes the input image. It outputs a grid of feature vectors



How the Attention Mechanism Works

Objective: To determine the most relevant image regions for generating each word in the caption.

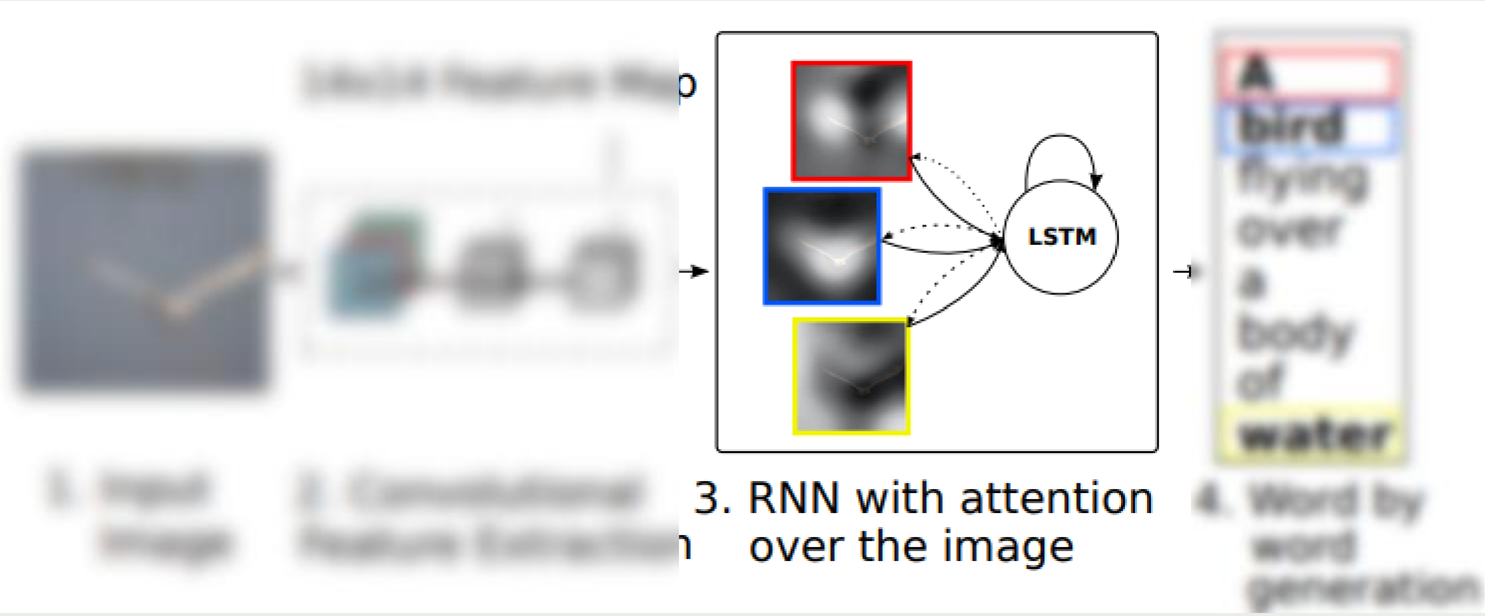
1. An image encoder (CNN) processes the input image. It outputs a grid of feature vectors
2. At each step, the **RNN** uses its hidden state to assess the relevance of each **14×14 Feature Map** region,..



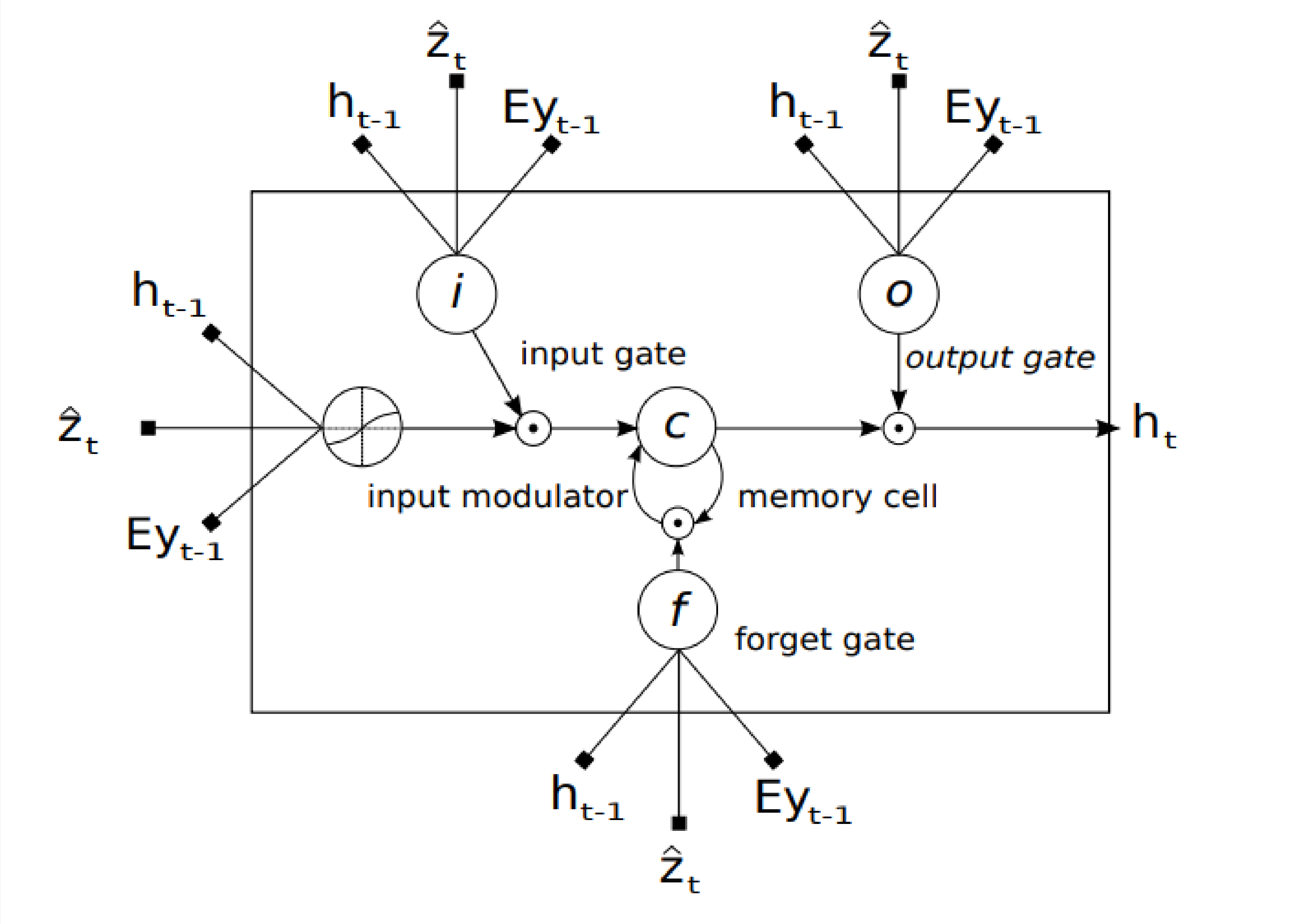
How the Attention Mechanism Works

Objective: To determine the most relevant image regions for generating each word in the caption.

1. An image encoder (CNN) processes the input image. It outputs a grid of feature vectors
2. At each step, the **RNN** uses its hidden state to assess the relevance of each **14×14 Feature Map** region, calculating **attention weights**. These weights then form a **context vector.**

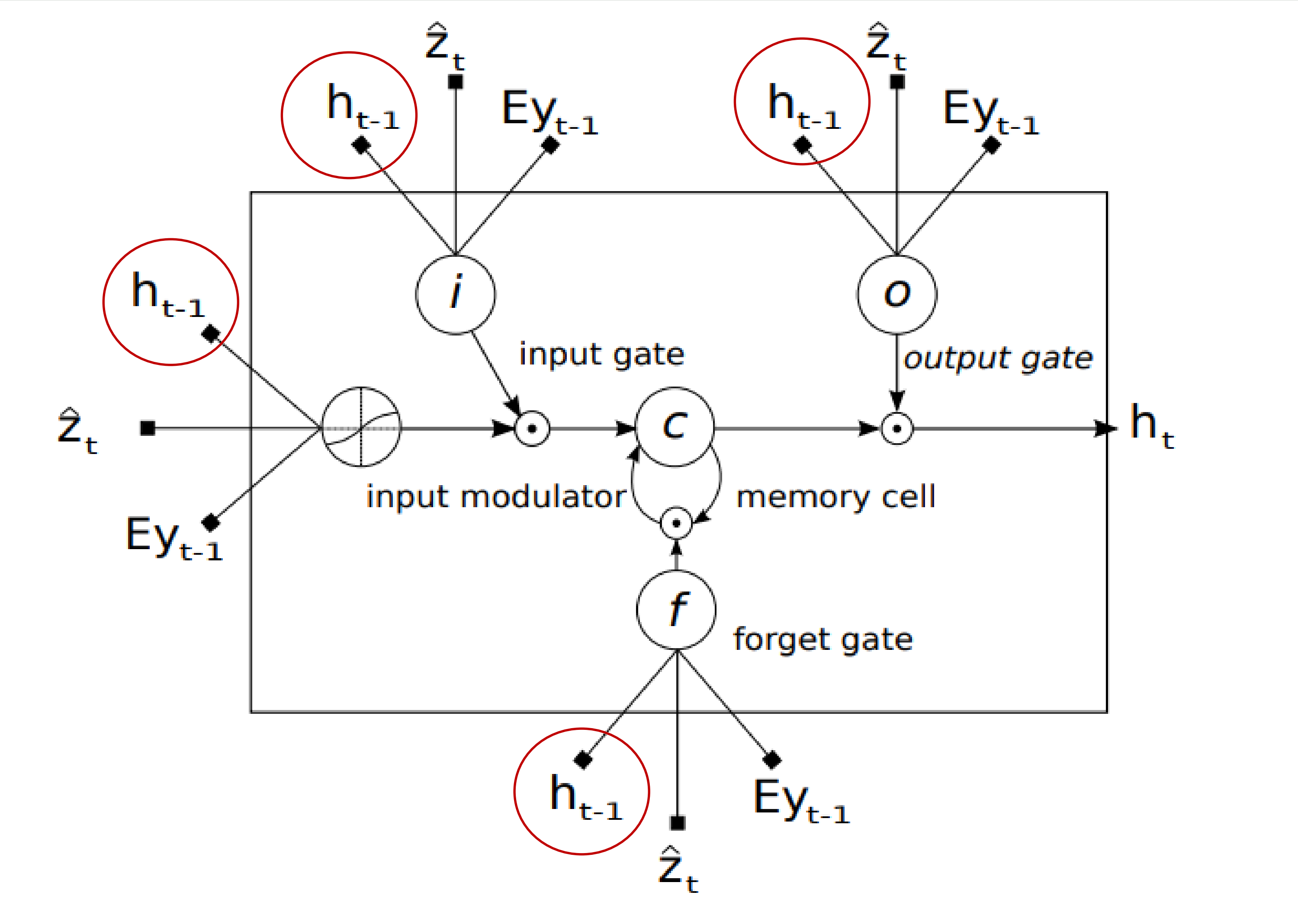


Decoder: The LSTM Architecture



Decoder: The LSTM Architecture

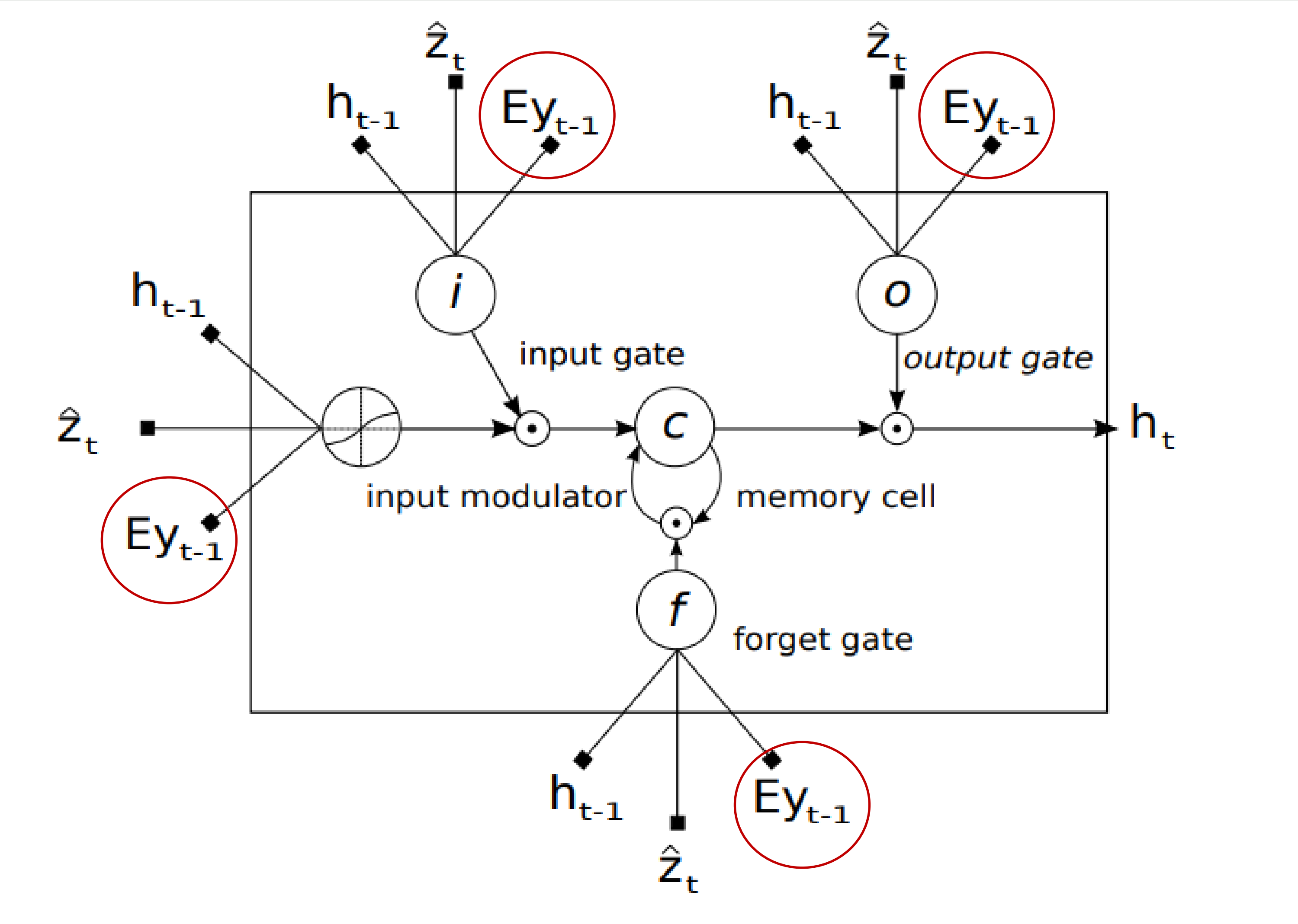
Previous Hidden State (h_{t-1}):
This is the LSTM's internal "memory" from the previous step.



Decoder: The LSTM Architecture

Previous Hidden State (h_{t-1}): This is the LSTM's internal "memory" from the previous step.

Previous Word Embedding (Ey_{t-1}): This is the numerical representation of the word the LSTM just predicted in the last step.

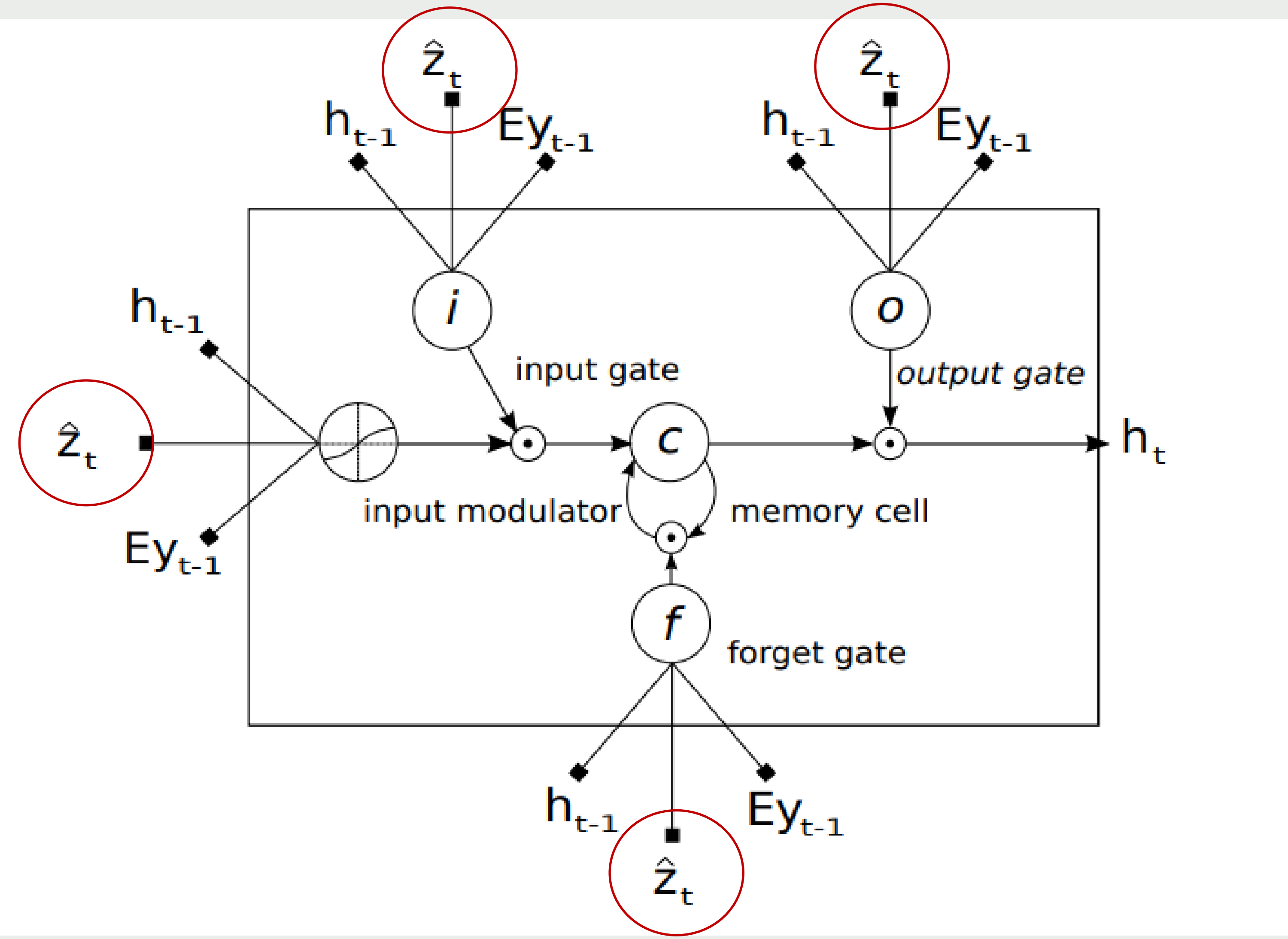


Decoder: The LSTM Architecture

Previous Hidden State (h_{t-1}): This is the LSTM's internal "memory" from the previous step.

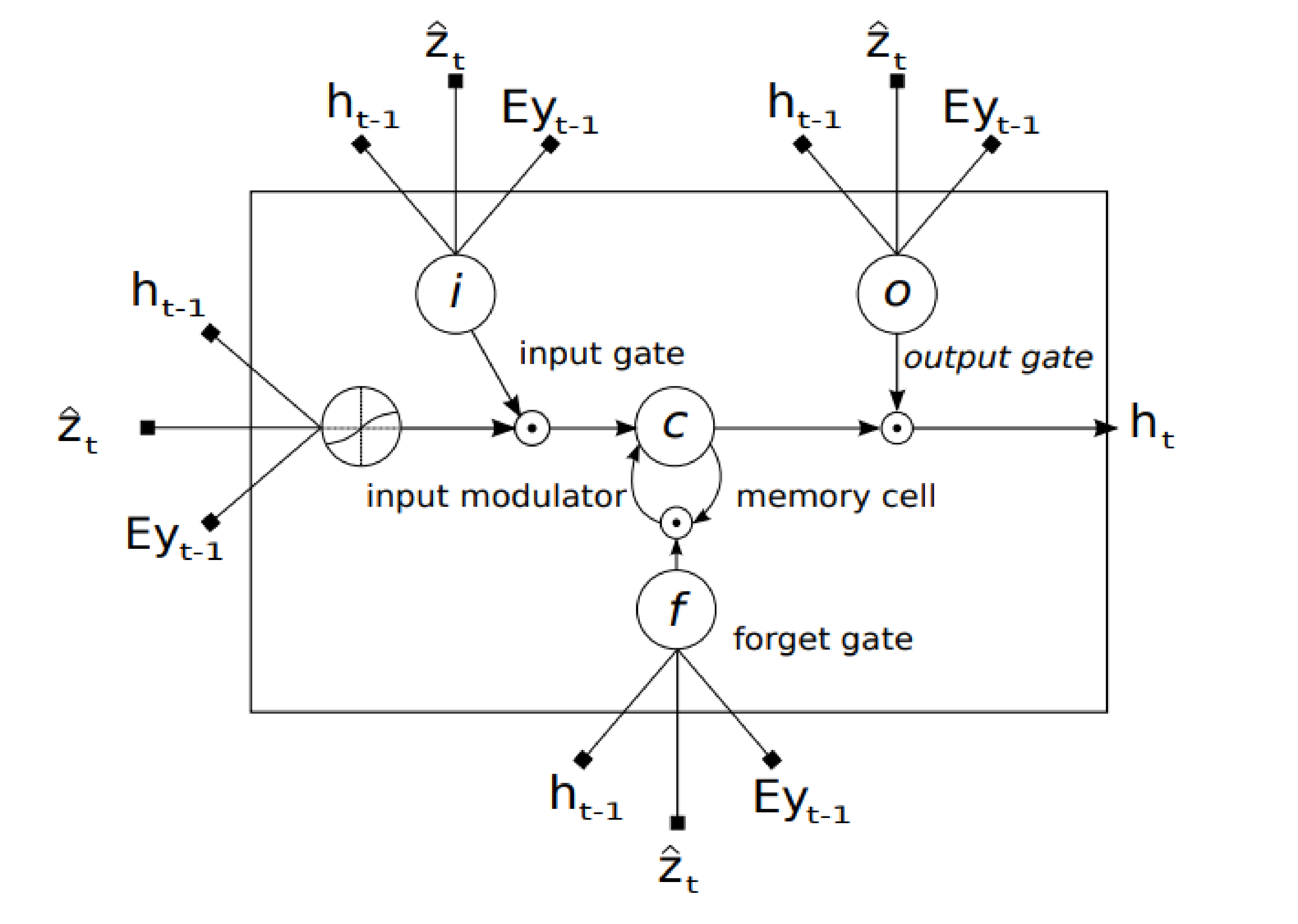
Previous Word Embedding (Ey_{t-1}): This is the numerical representation of the word the LSTM just predicted in the last step.

Context Vector ($\hat{Z}_t$): This is the important visual summary provided by the attention mechanism.



Decoder: The LSTM Architecture

The LSTM has special internal "gates" (like input gate, forget gate, output gate) and a "memory cell" (c)



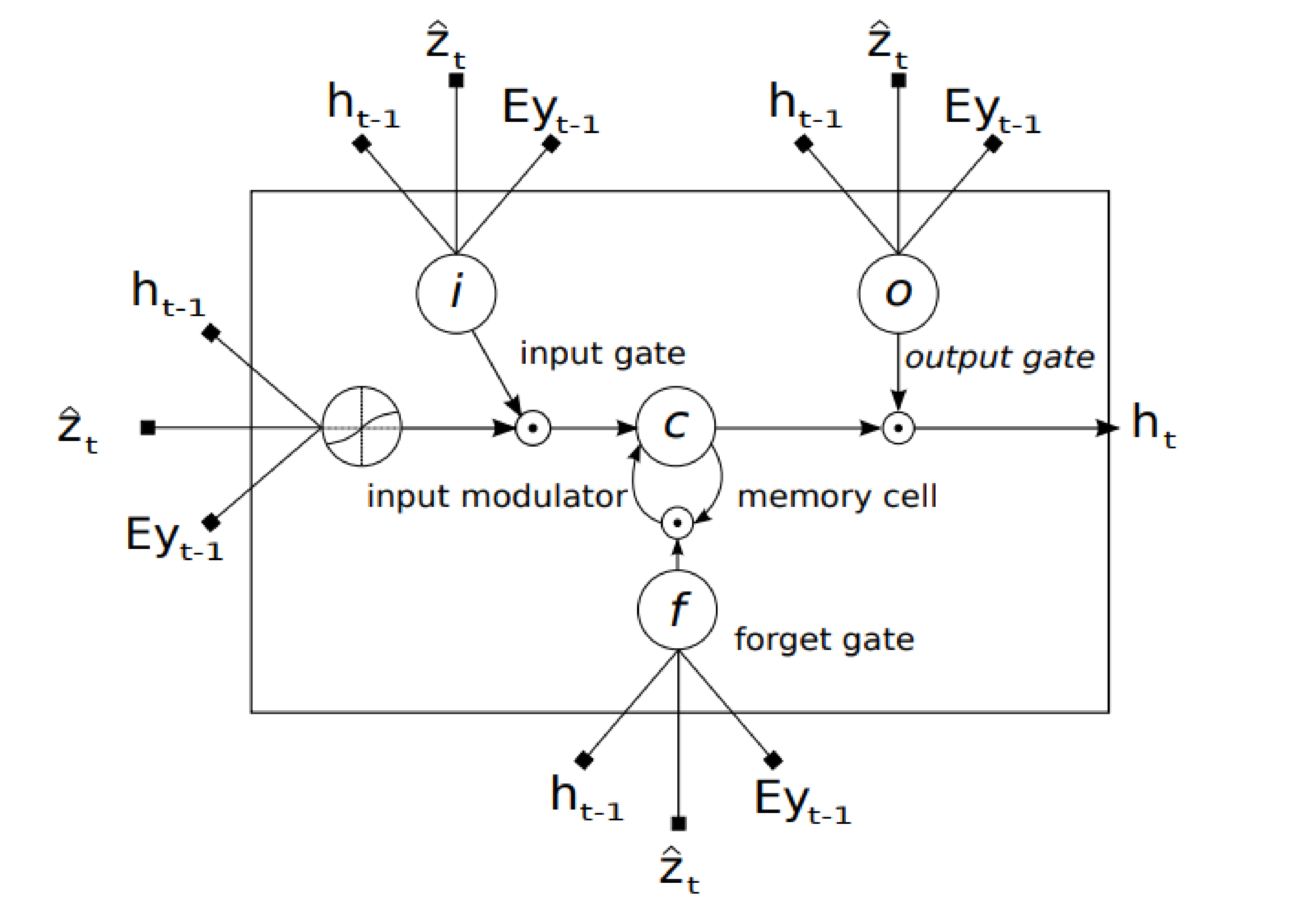
Decoder: The LSTM Architecture

The LSTM has special internal "gates" (like input gate, forget gate, output gate) and a "memory cell" (c) :

Forget Gate (f): Decides what to throw away from the *previous* cell state.

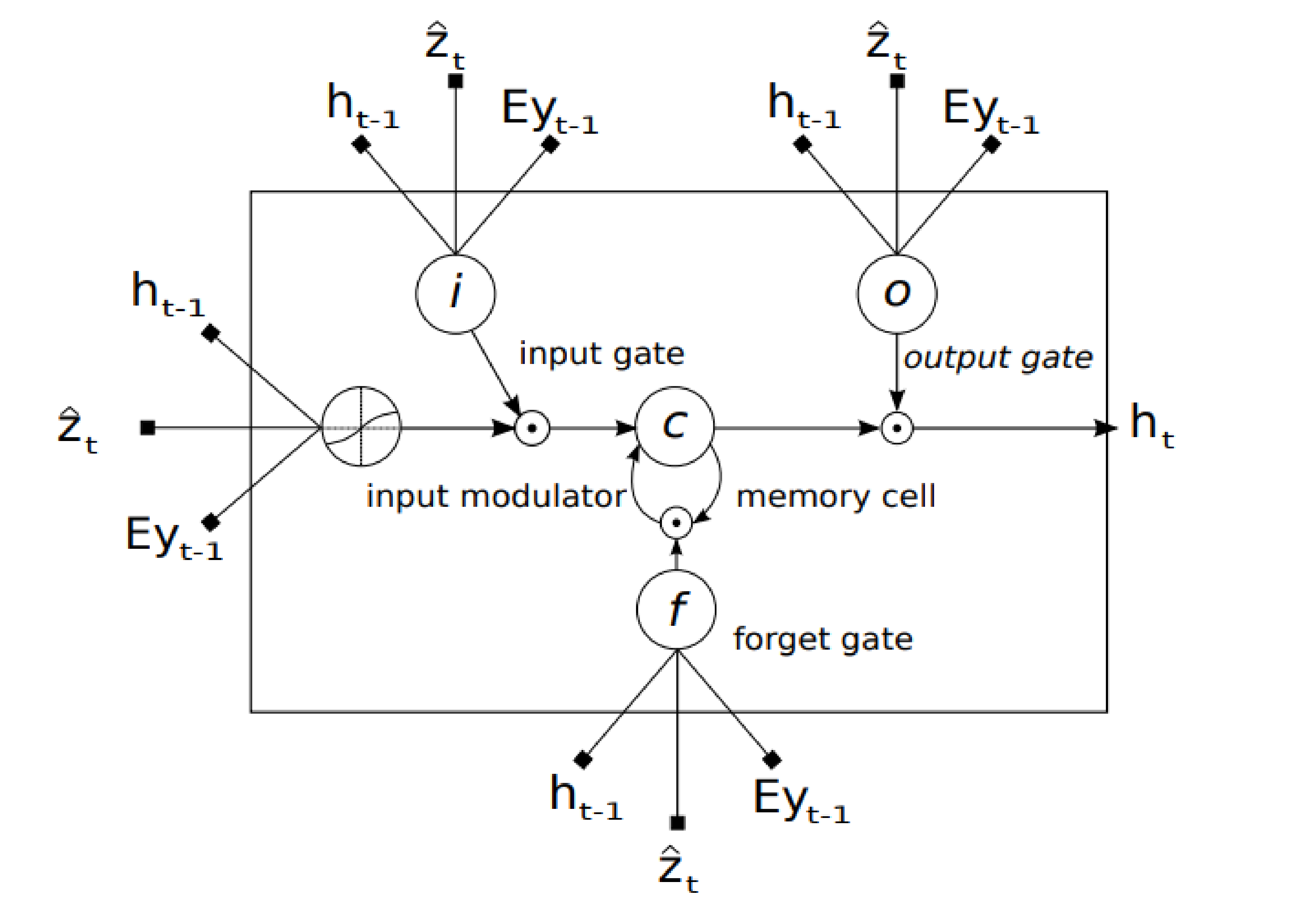
Input Gate (i): Decides what *new information* to store in the cell state.

Output Gate (o): Decides what parts of the *current* cell state to output as the hidden state.



Decoder: The LSTM Architecture

Output and Prediction: Based on these internal calculations, the LSTM produces a new **hidden state (h_t)**. This h_t contains all the updated information needed.

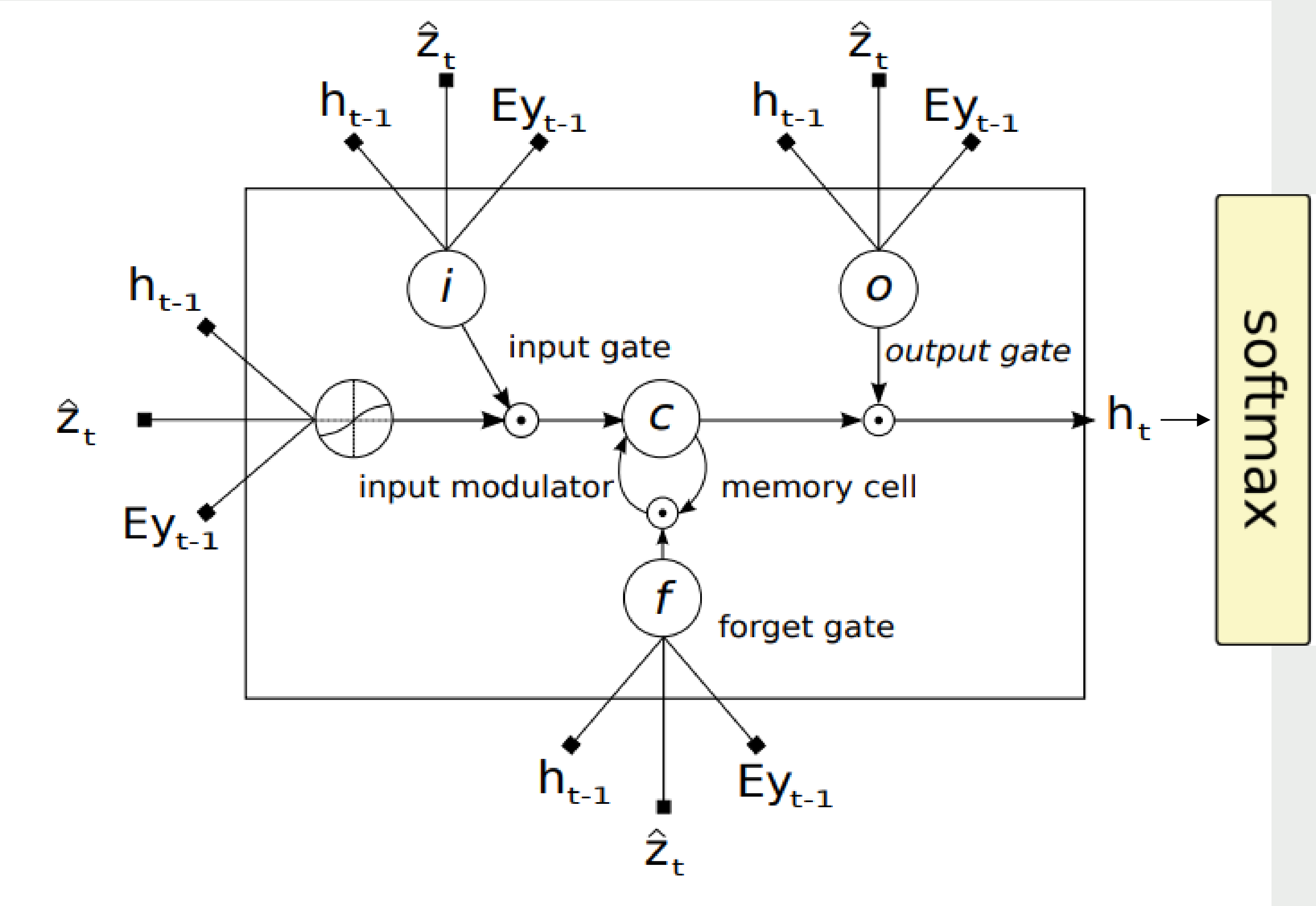


Decoder: The LSTM Architecture

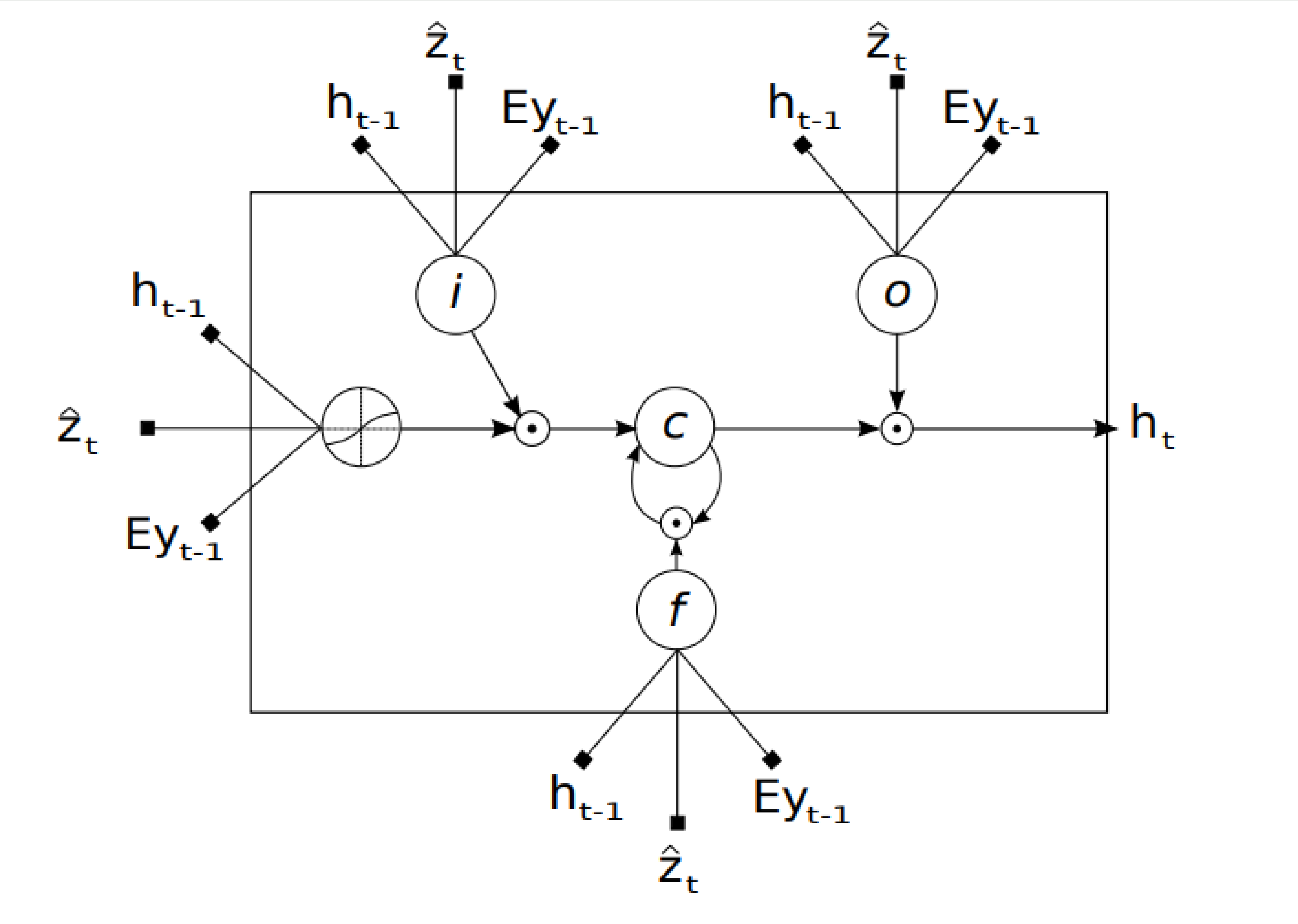
Output and Prediction: Based on these internal calculations, the LSTM produces a new **hidden state (h_t)**. This h_t contains all the updated information needed.

This h_t is then used by another part of the model to predict the next word in the caption.

The LSTM then uses h_t as the h_{t-1} for the next step, repeating the process until the full caption is completed.



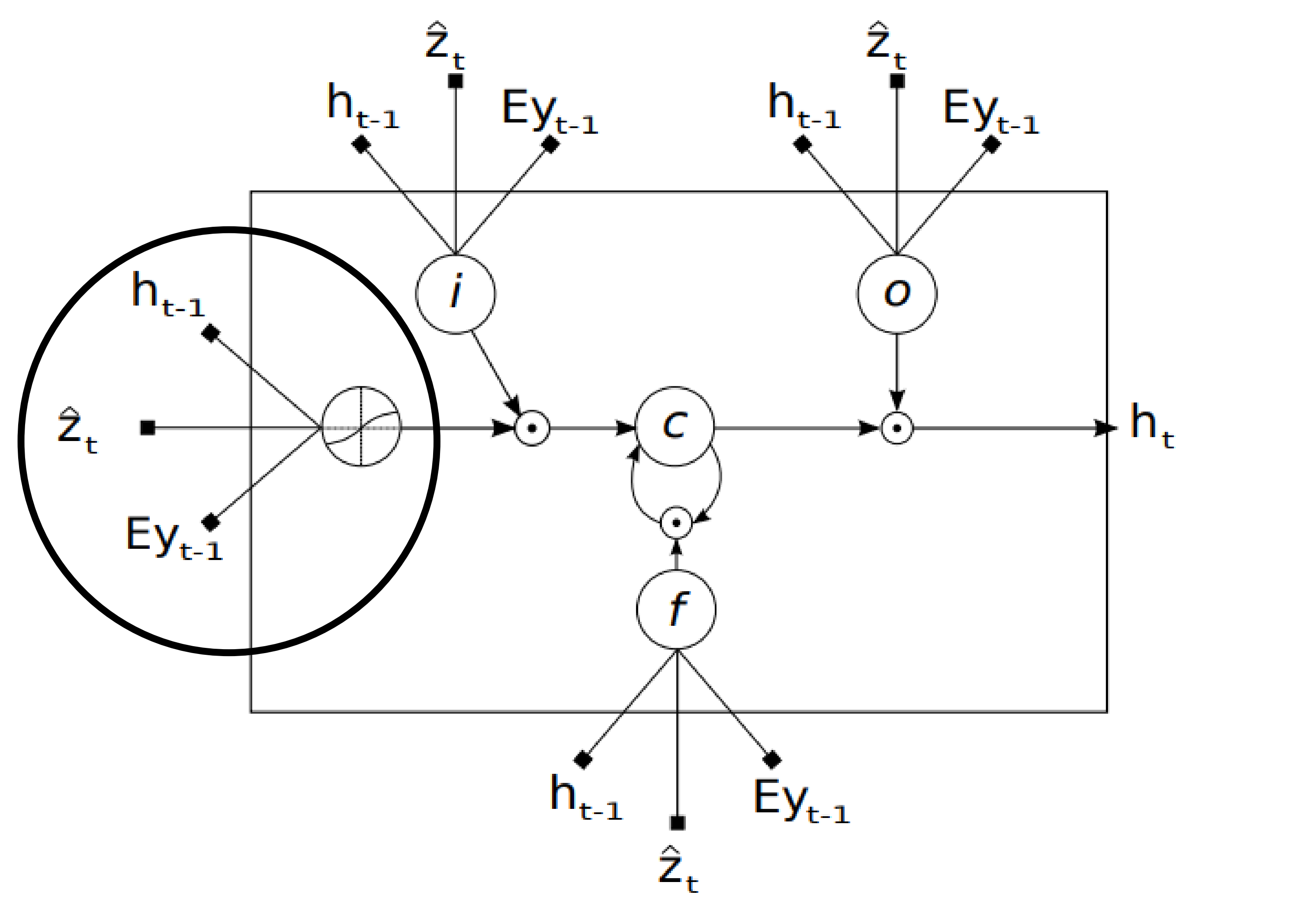
Decoder: The LSTM Architecture



Decoder: The LSTM Architecture

1. **Preparing New Information to Potentially Add to Memory:**

a . Input Modulator:
Processes the inputs (using a tanh activation function) to create a vector of candidate values

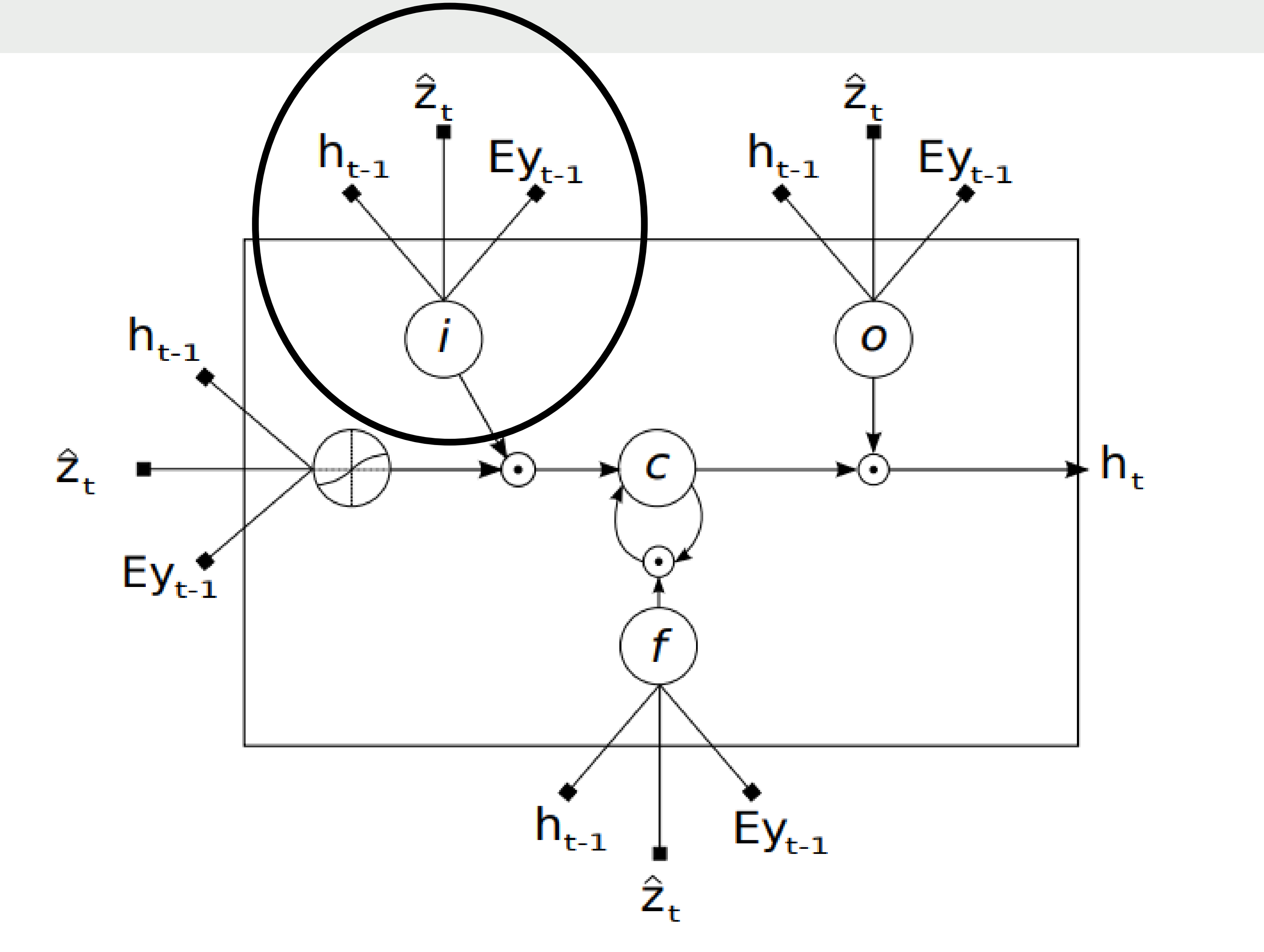


Decoder: The LSTM Architecture

1. **Preparing New Information to Potentially Add to Memory:**

a . Input Modulator:
Processes the inputs (using a tanh activation function) to create a vector of candidate values

b. Input Gate (i):
The input gate calculates a filter, values are between 0 and 1.



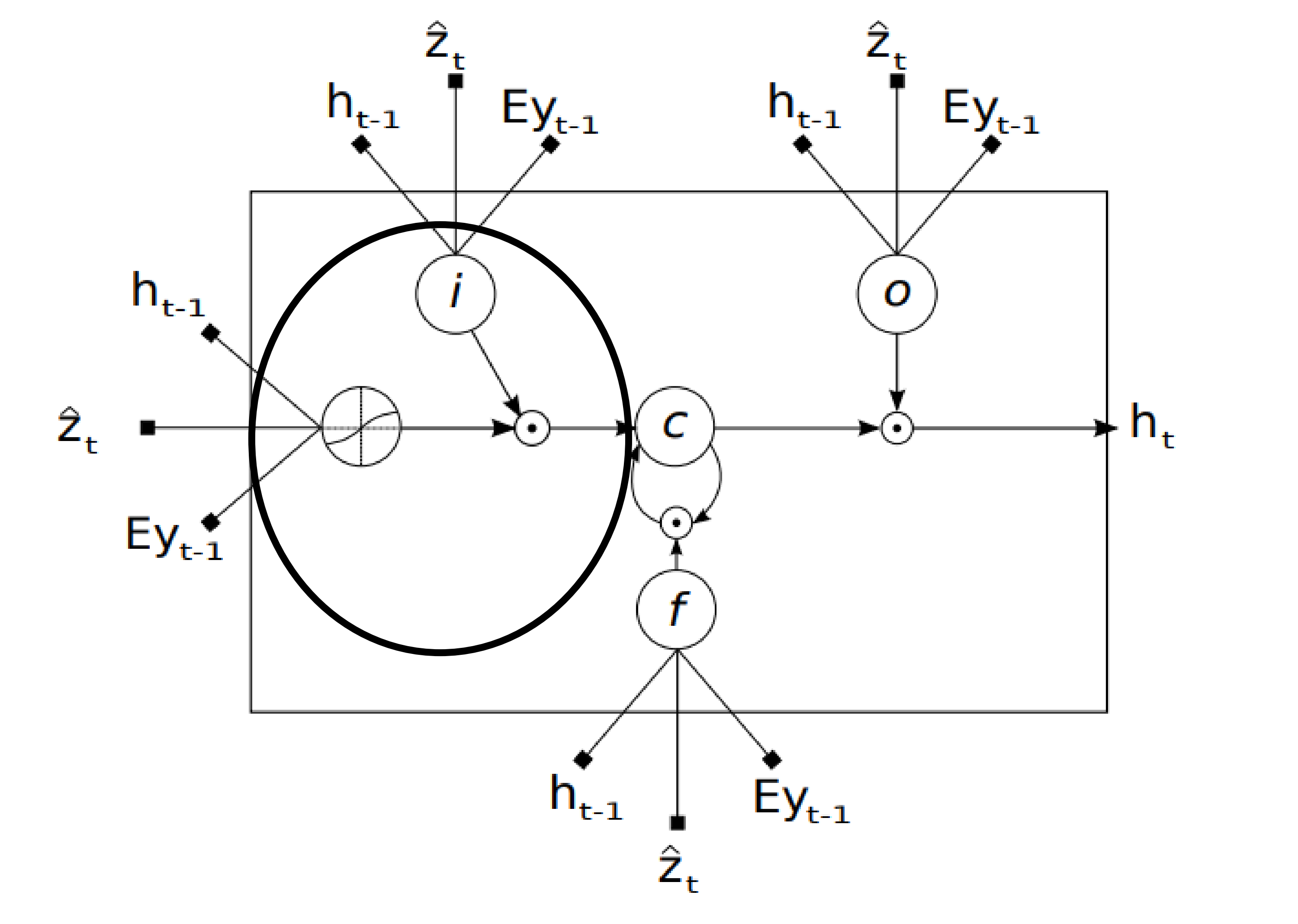
Decoder: The LSTM Architecture

1. **Preparing New Information to Potentially Add to Memory:**

a . Input Modulator:
Processes the inputs (using a tanh activation function) to create a vector of candidate values

b. Input Gate (i):
The input gate calculates a filter, values are between 0 and 1.

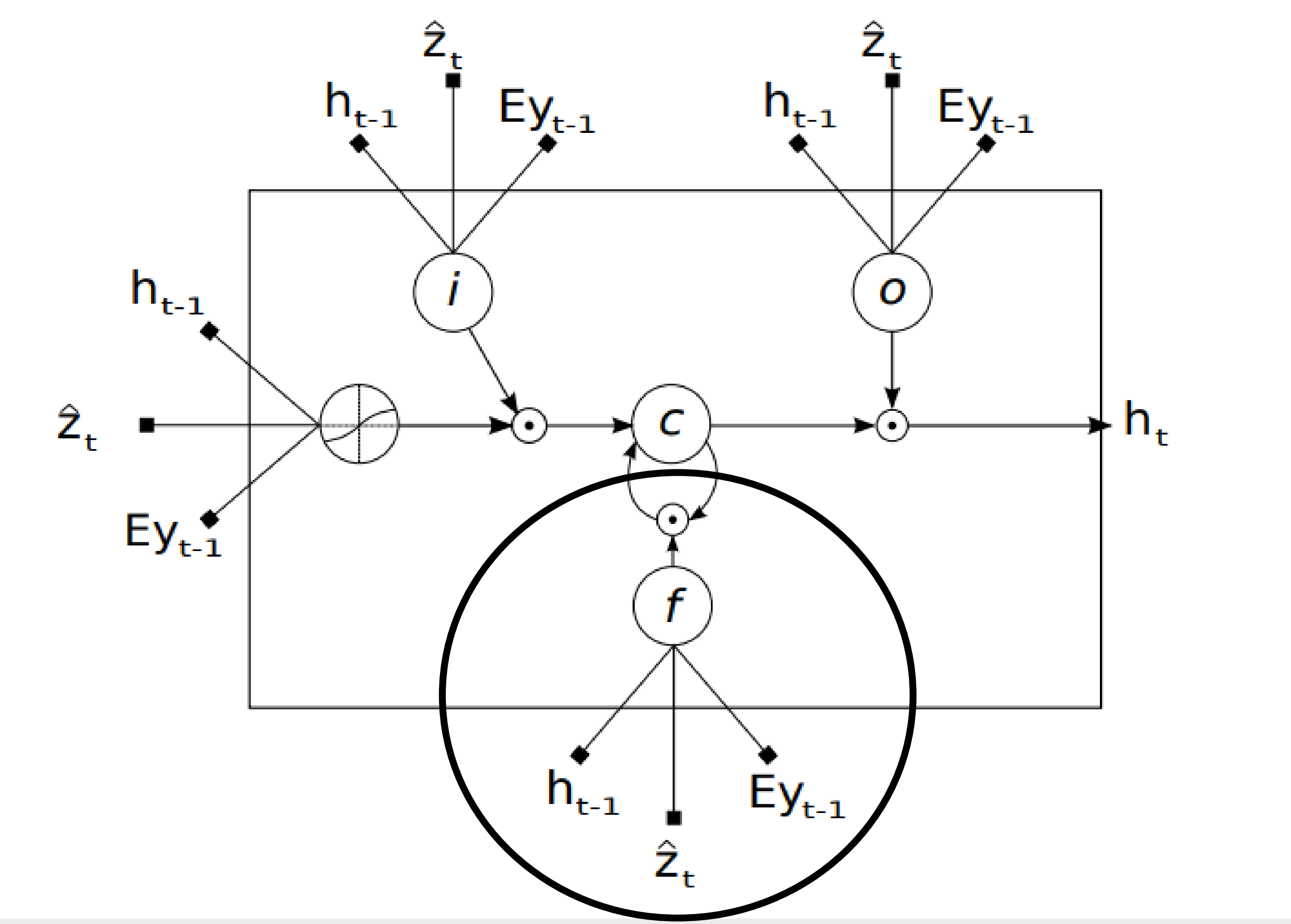
The output of the input modulator (is then element-wise multiplied by the output of the input gate to get new information



Decoder: The LSTM Architecture

2. Deciding What to Forget from Old Memory:

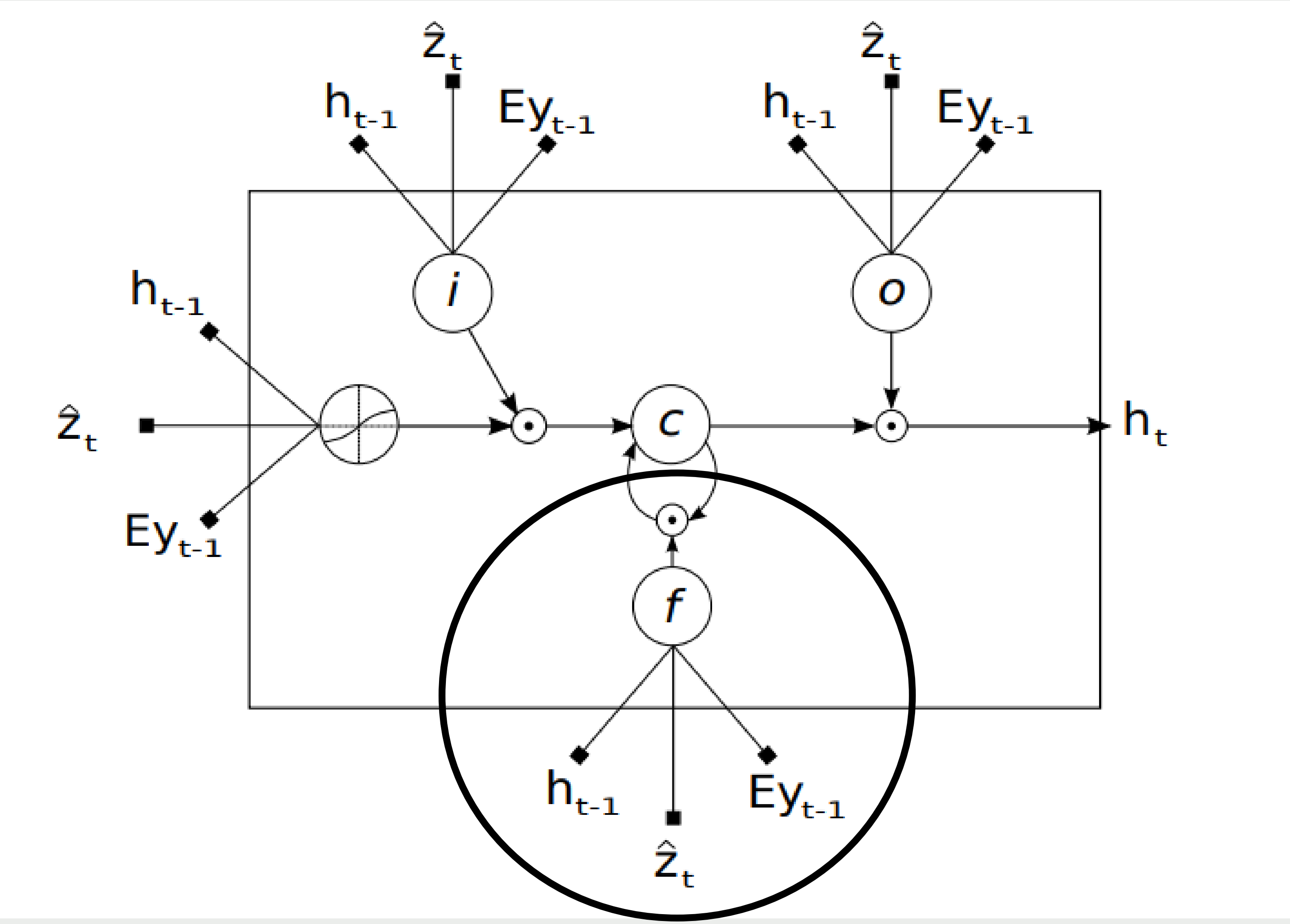
Forget Gate (f):
The same three core inputs are also fed into the forget gate. The forget gate calculates another filter.



Decoder: The LSTM Architecture

2. Deciding What to Forget from Old Memory:

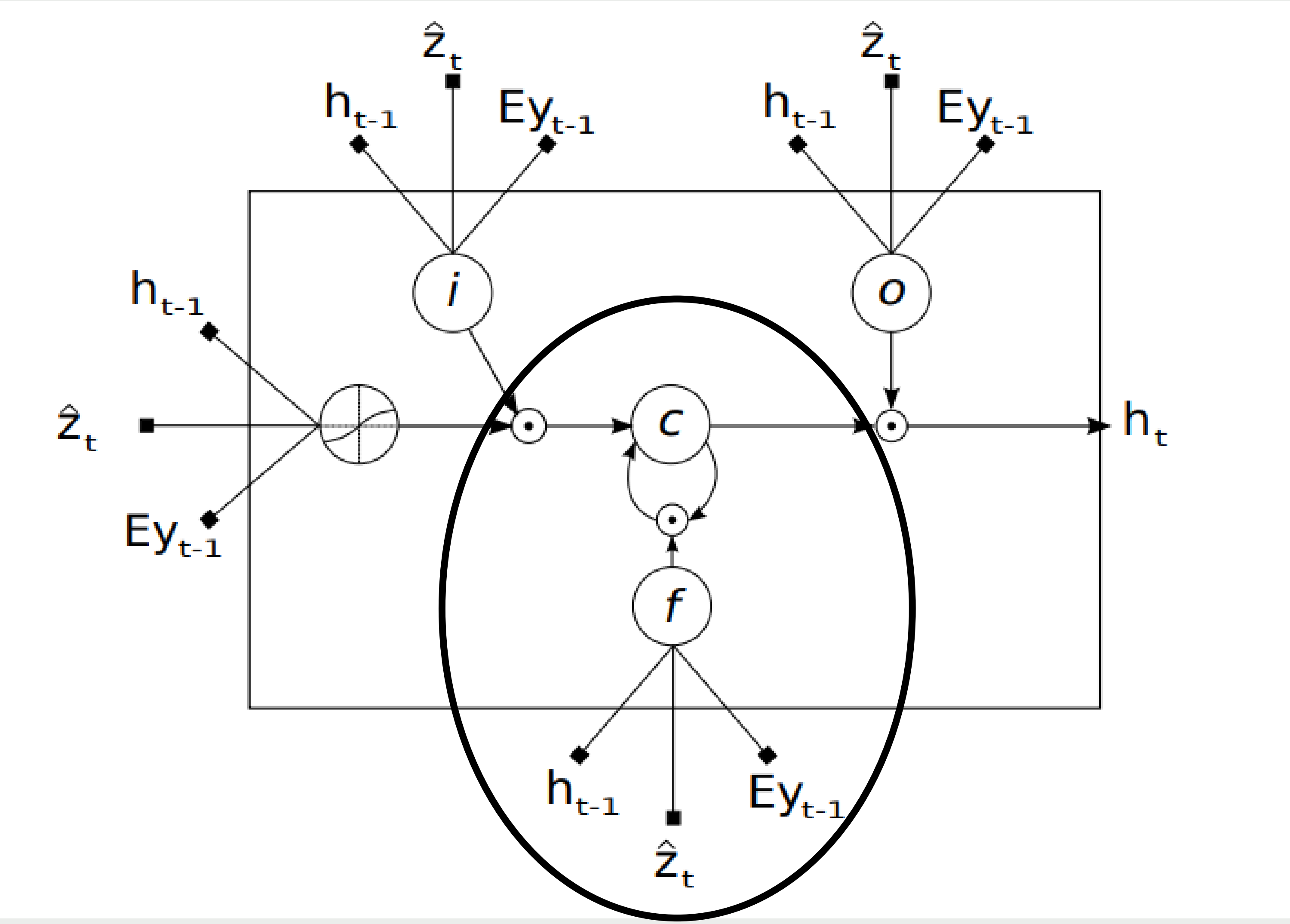
Forget Gate (f):
The same three core inputs are also fed into the forget gate. The forget gate calculates another filter.



Decoder: The LSTM Architecture

3. Updating the Main Memory Cell ©

First, the old memory from the previous time step is element-wise multiplied by the output of the forget gate .This is the "forgetting" step.

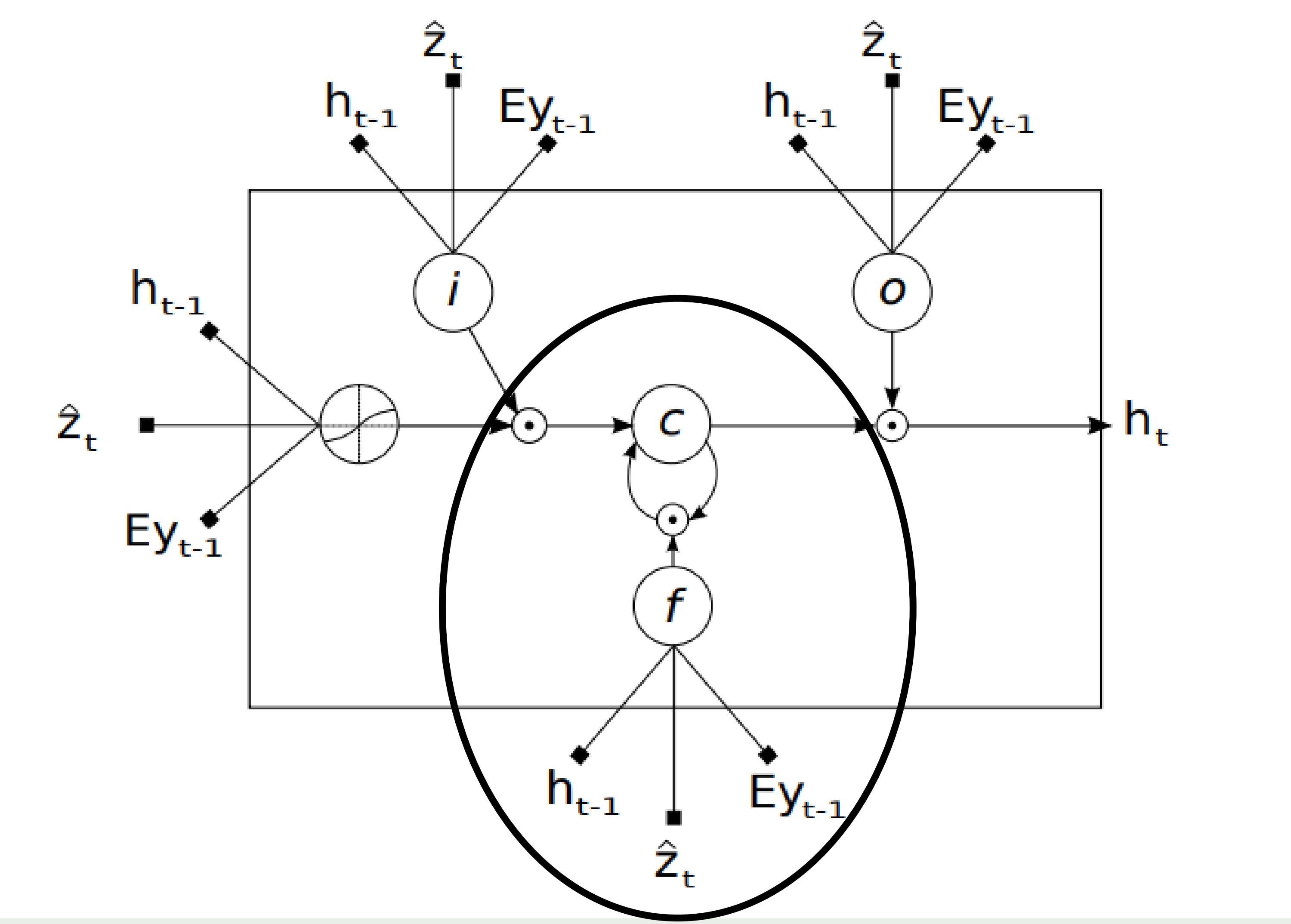


Decoder: The LSTM Architecture

3. Updating the Main Memory Cell ©

First, the old memory from the previous time step is element-wise multiplied by the output of the forget gate .This is the "forgetting" step.

Then, the filtered new information is added to the result of the forgetting step.



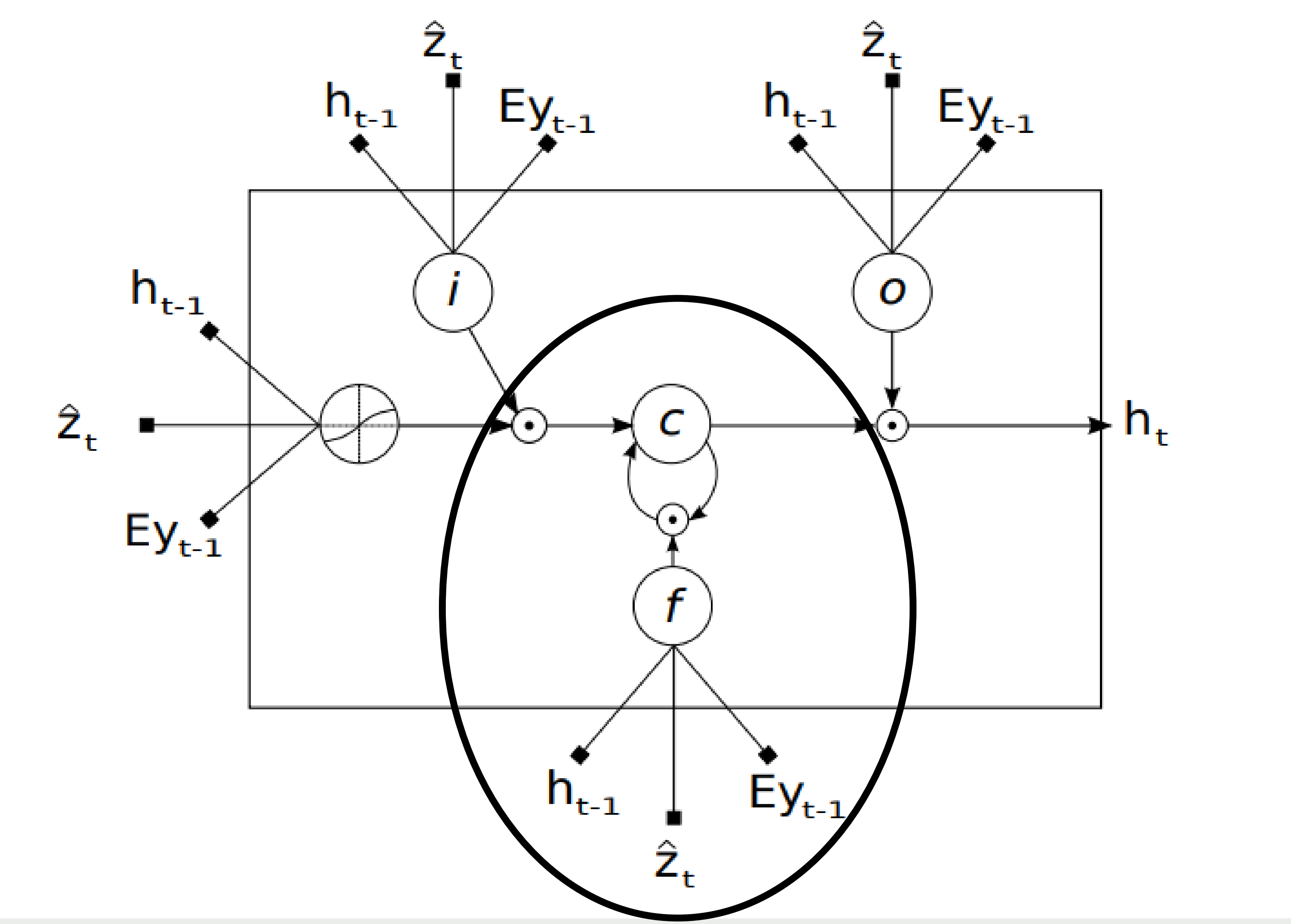
Decoder: The LSTM Architecture

3. Updating the Main Memory Cell ©

First, the old memory from the previous time step is element-wise multiplied by the output of the forget gate .This is the "forgetting" step.

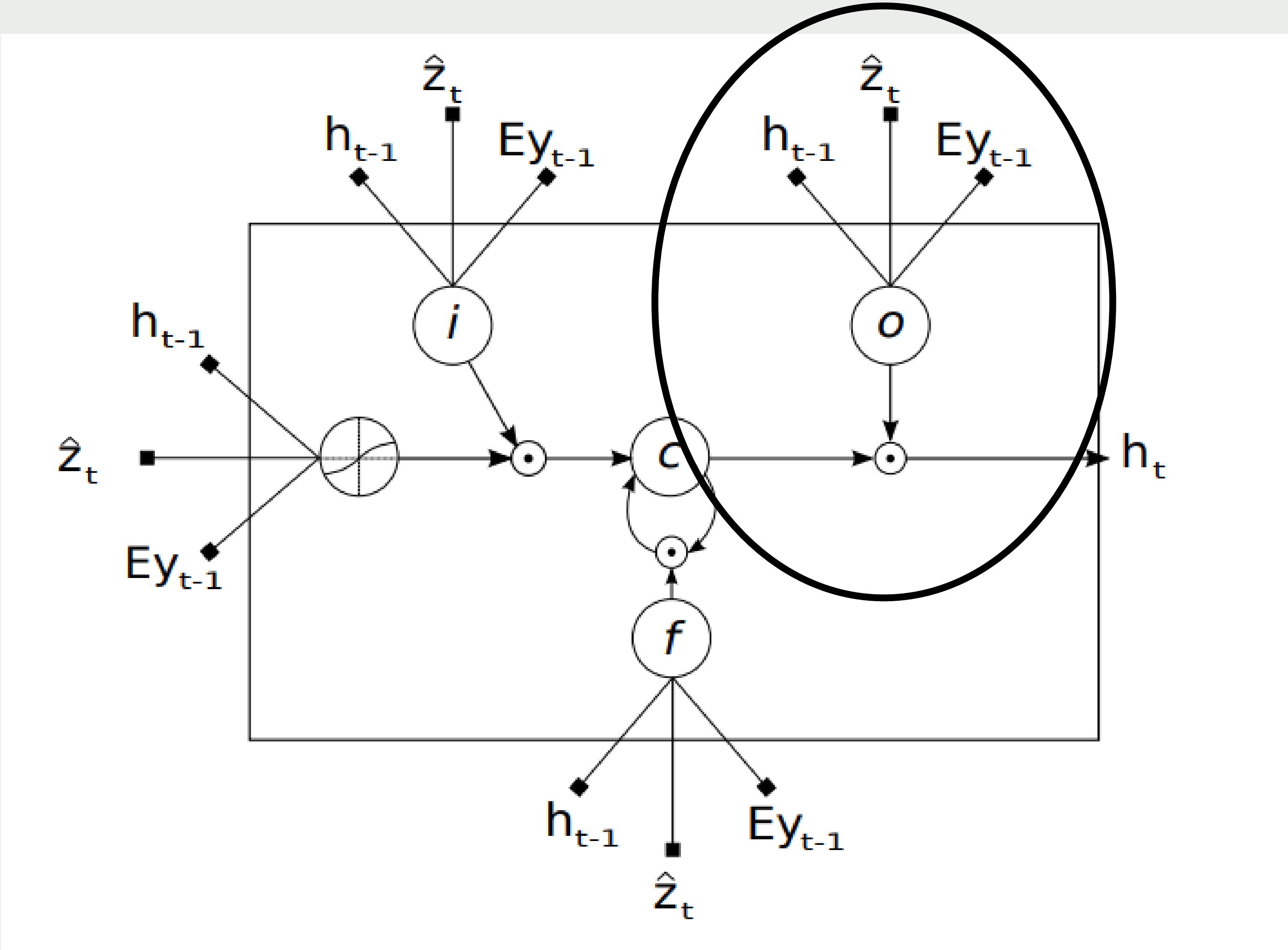
Then, the filtered new information is added to the result of the forgetting step.

This sum becomes the new, updated state of the memory cell for the current time step, C_T.
So:
C_T = (output of forget gate * C_{T-1}) + (output of input modulator * output of input gate)



Decoder: The LSTM Architecture

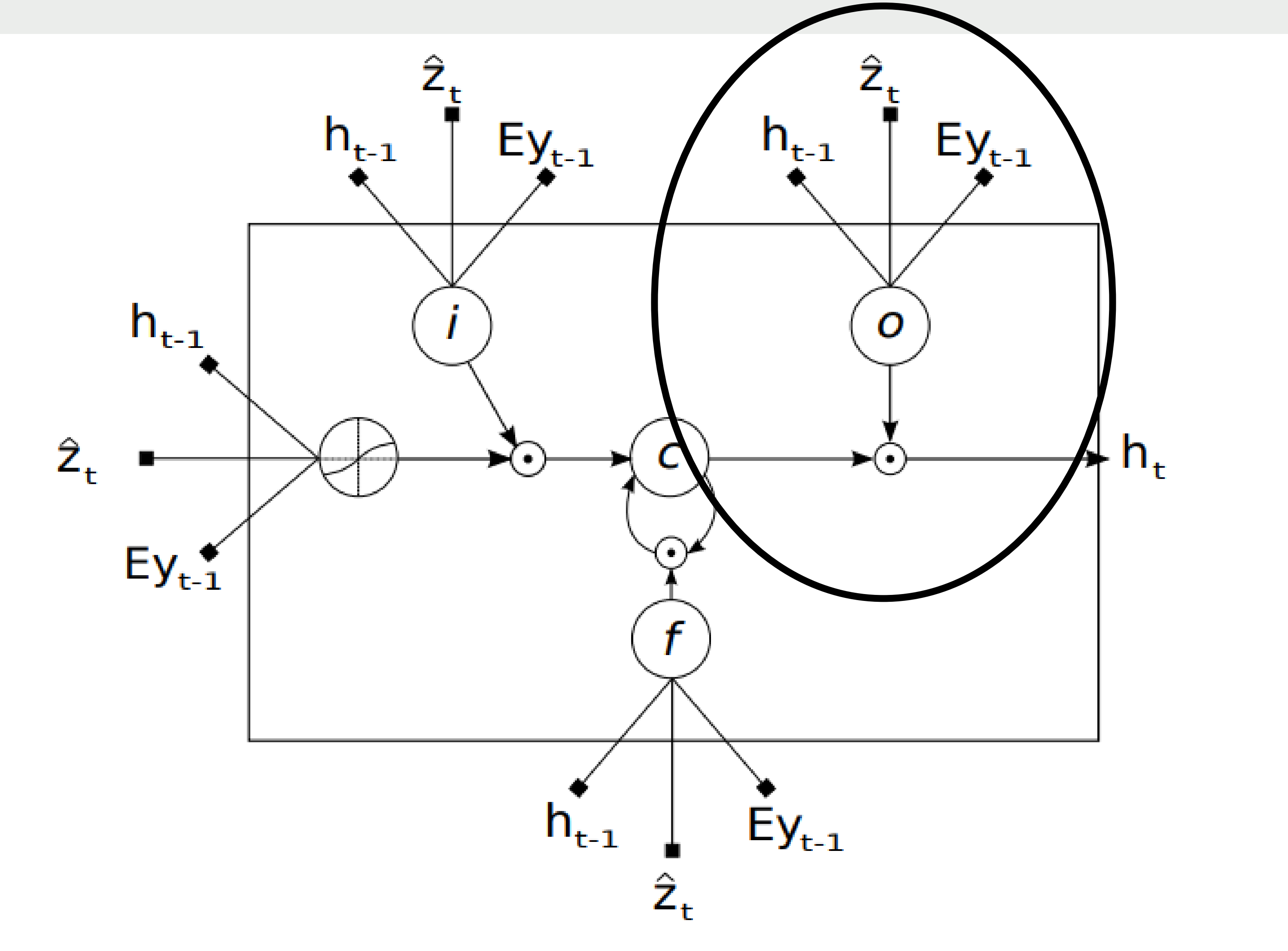
4. Preparing the Output for this Time Step:
a. Output Gate :
The inputs are put into the "output gate“ and then calculates a filter.



Decoder: The LSTM Architecture

4. Preparing the Output for this Time Step:
a. Output Gate :
The inputs are put into the "output gate“ and then calculates a filter.

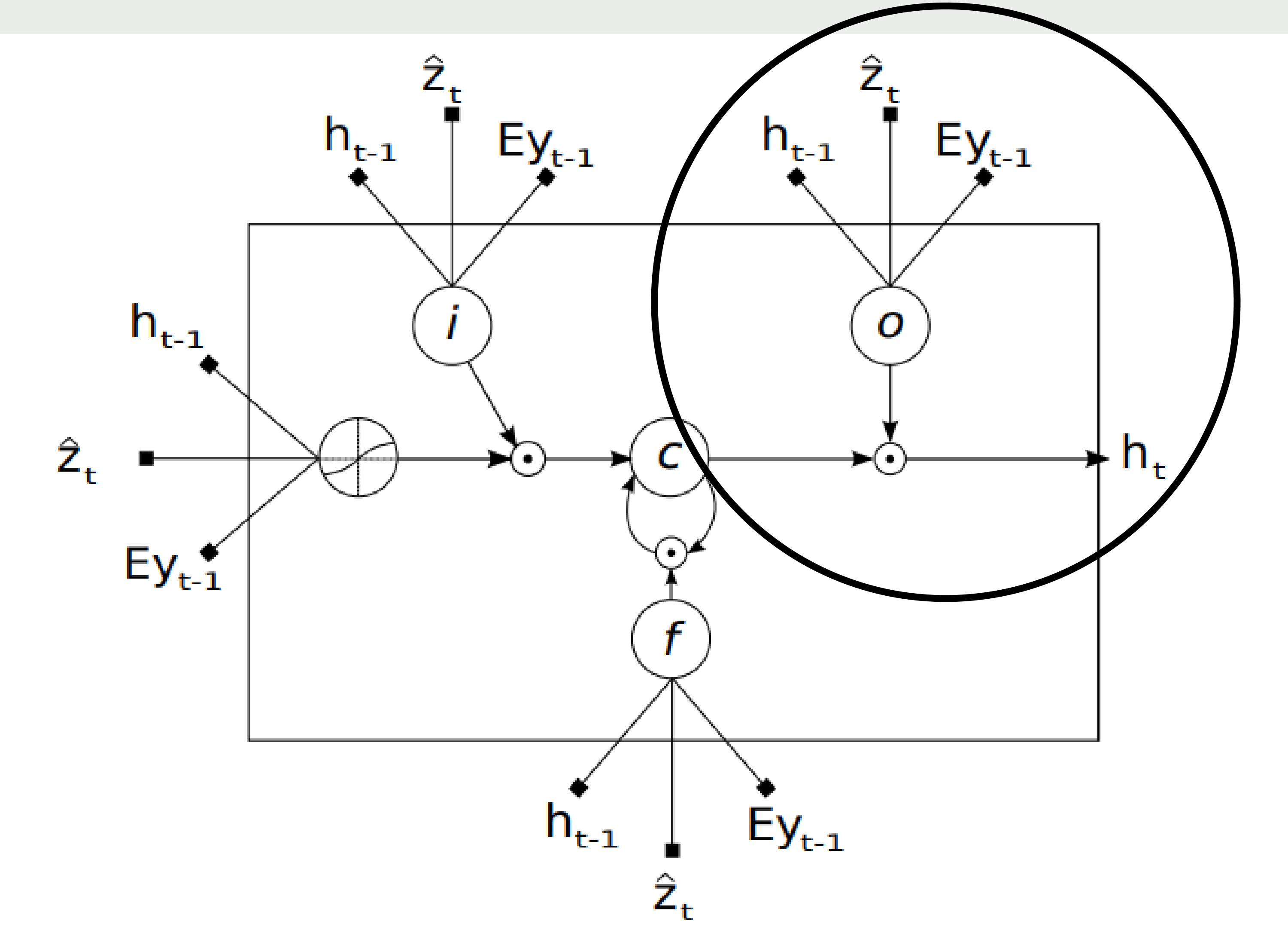
b. Processing the Updated Memory for Output:
The content of the updated memory cell is passed through a tanh function. This scales the memory content to a range between -1 and 1.



Decoder: The LSTM Architecture

4. Preparing the Output for this Time Step:
a. Output Gate :
The inputs are put into the "output gate“ and then calculates a filter.

b. Processing the Updated Memory for Output:
The content of the updated memory cell is passed through a tanh function. This scales the memory content to a range between -1 and 1.
c. Generating the New Hidden State (h_t):
The tanh-processed memory is then element-wise multiplied by the output of the output gate
The result of this multiplication is the new hidden state (h_t). This h_t is the final output of the LSTM cell for the current time step t.

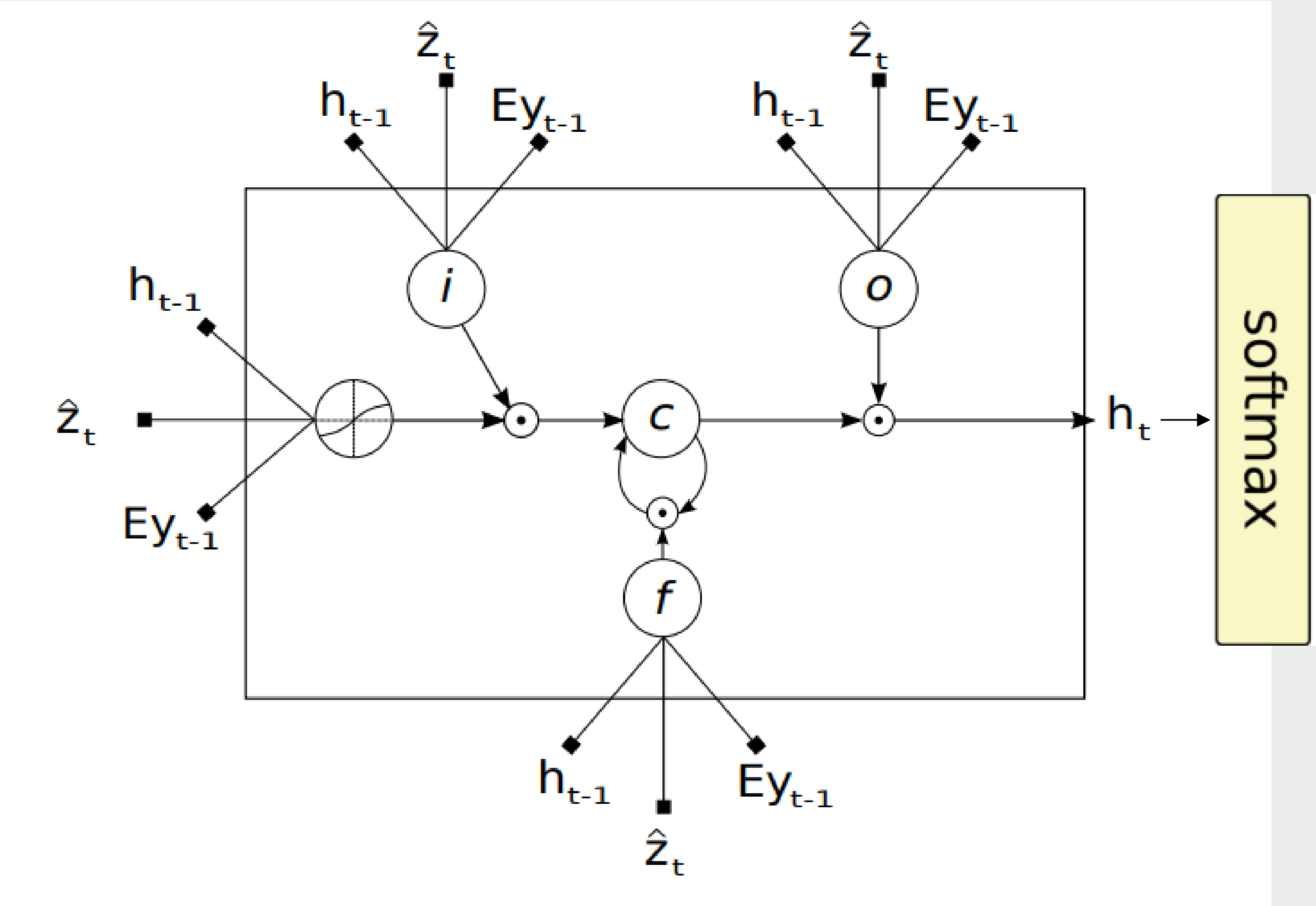


Decoder: The LSTM Architecture

What h_t is used for:

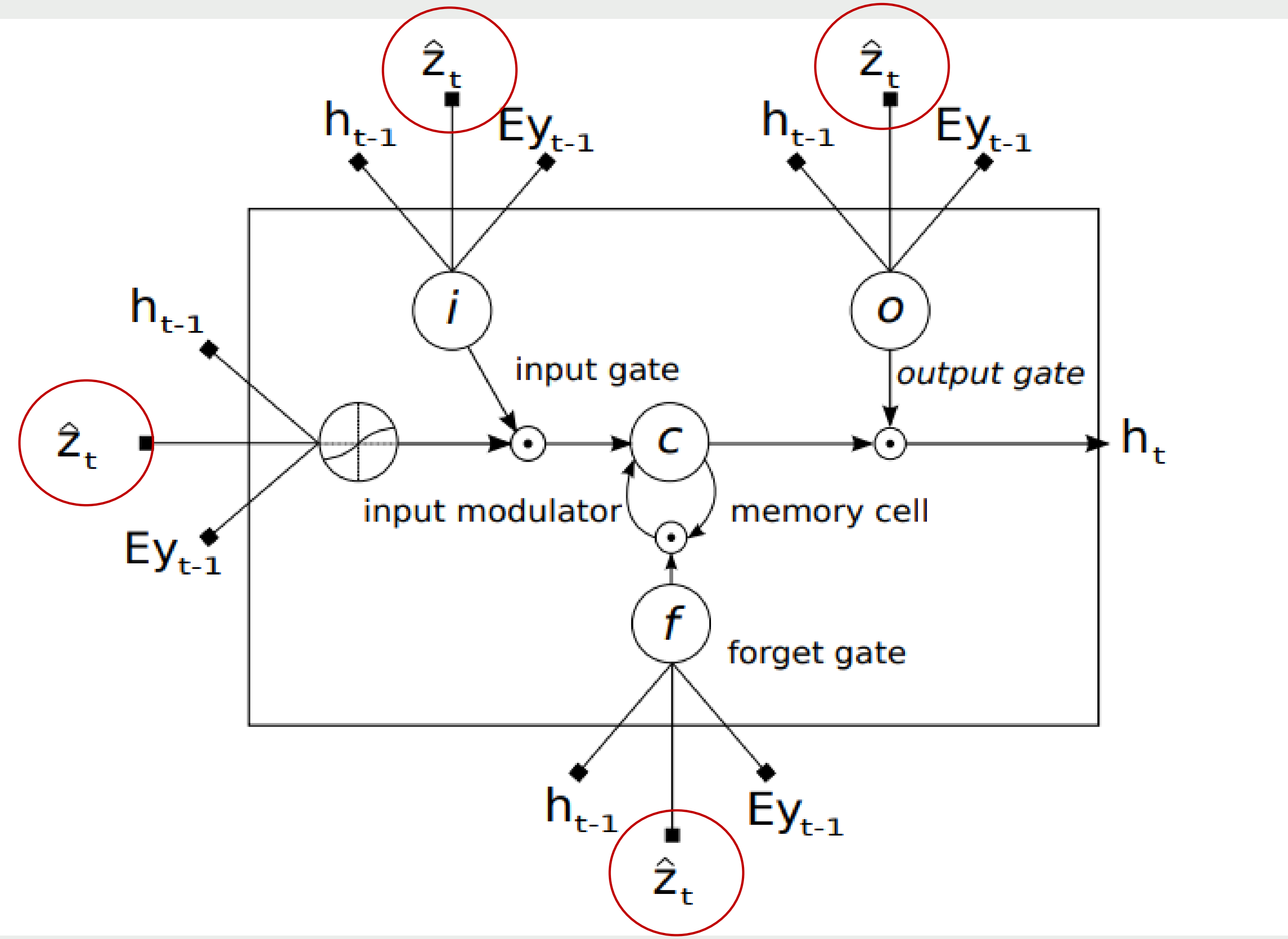
It will be used to predict the actual word $\boldsymbol{y_t}$ for the current position in the caption.

It will become $\boldsymbol{h_{t-1}}$ for the next time step, when the LSTM tries to generate the next word ($\boldsymbol{y_{t+1}}$).

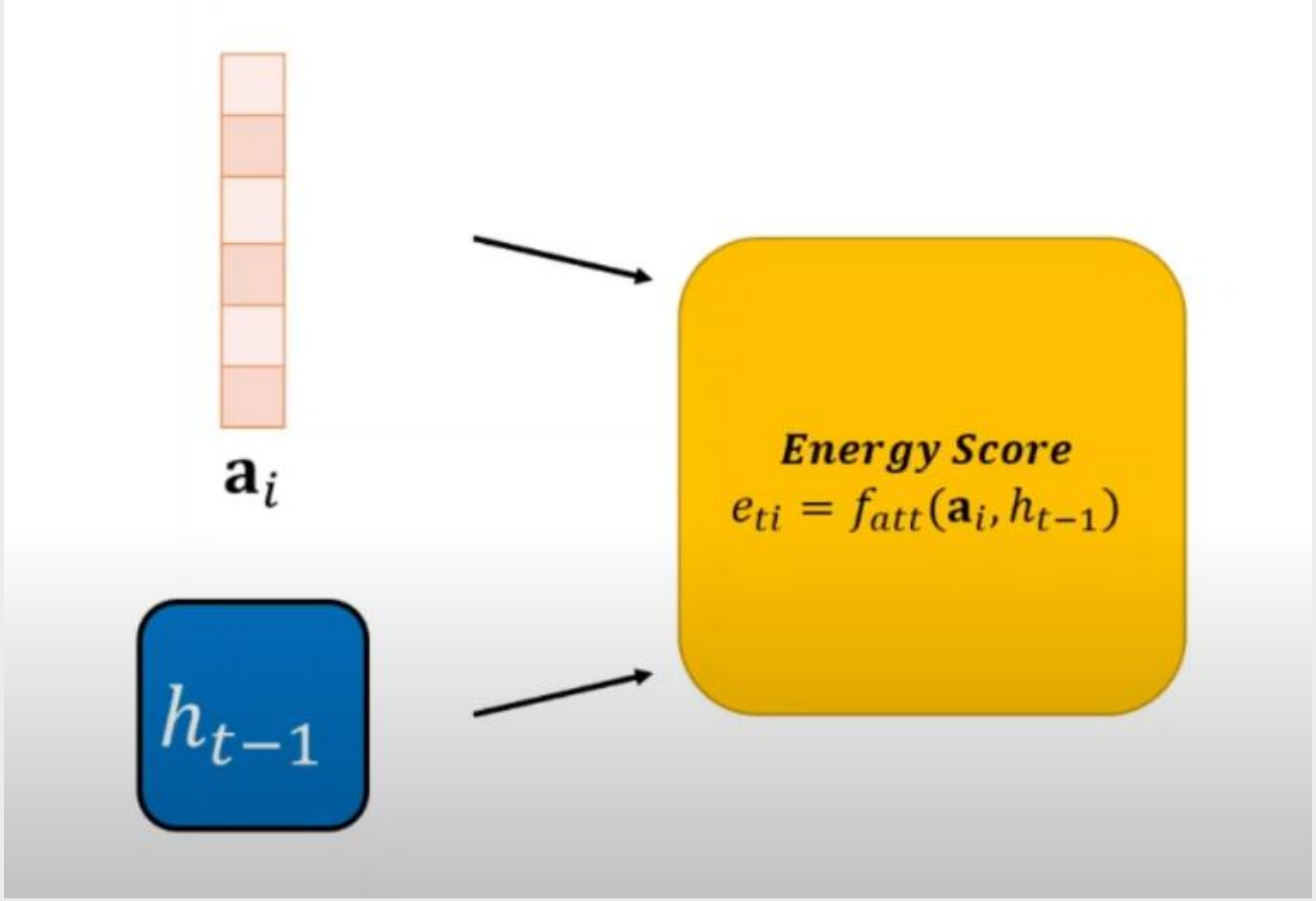


Decoder: The LSTM Architecture

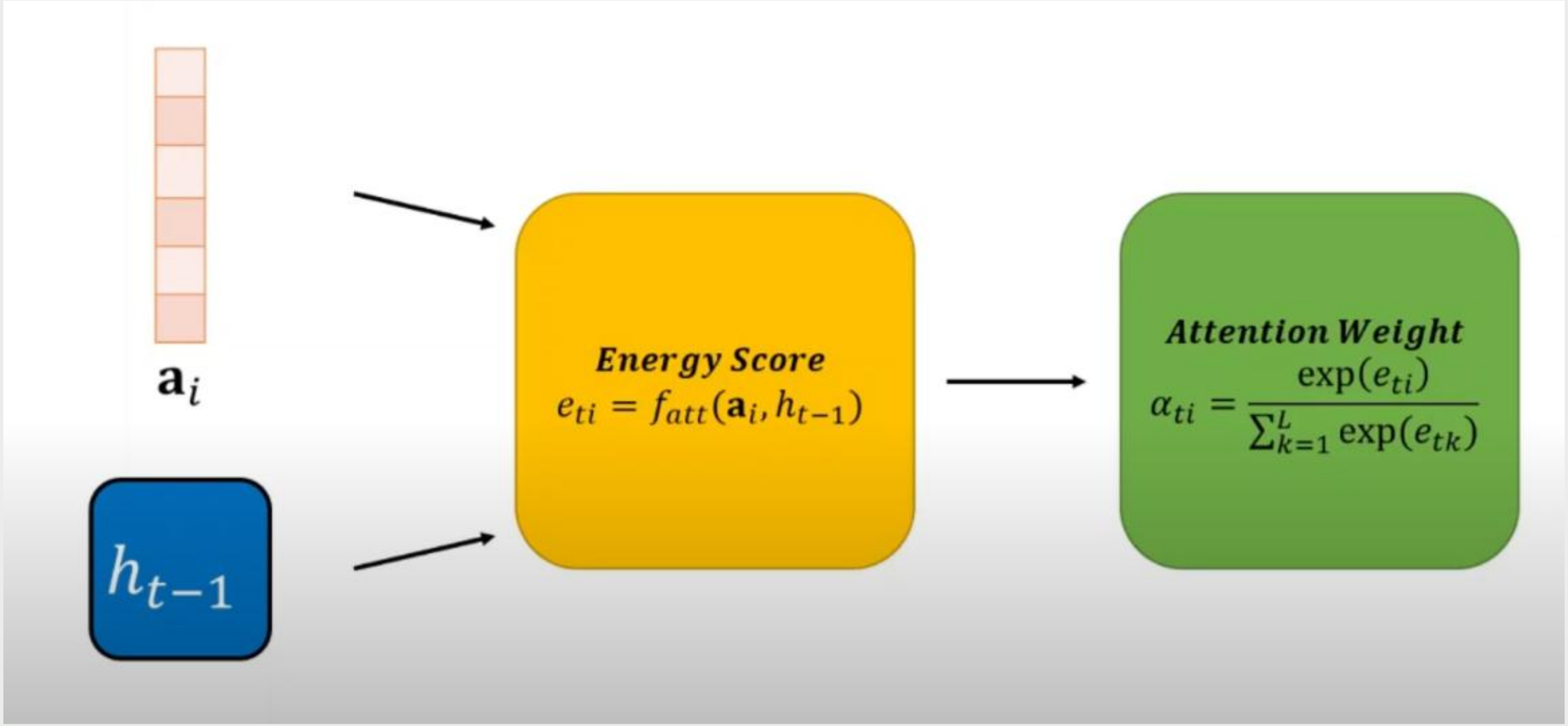
Context Vector ($\hat{Z}_t$): This is the important visual summary provided by the attention mechanism.



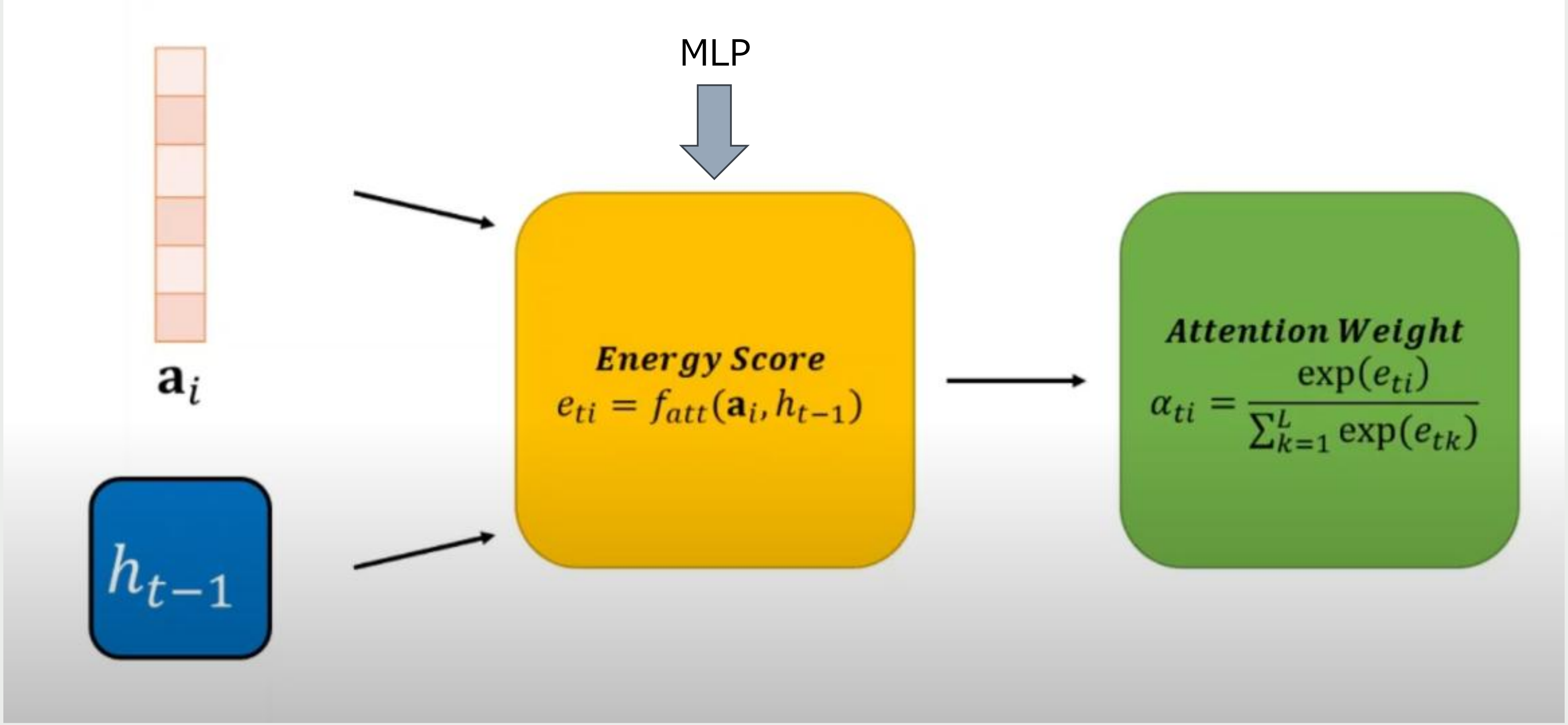
How is context vector calculated?



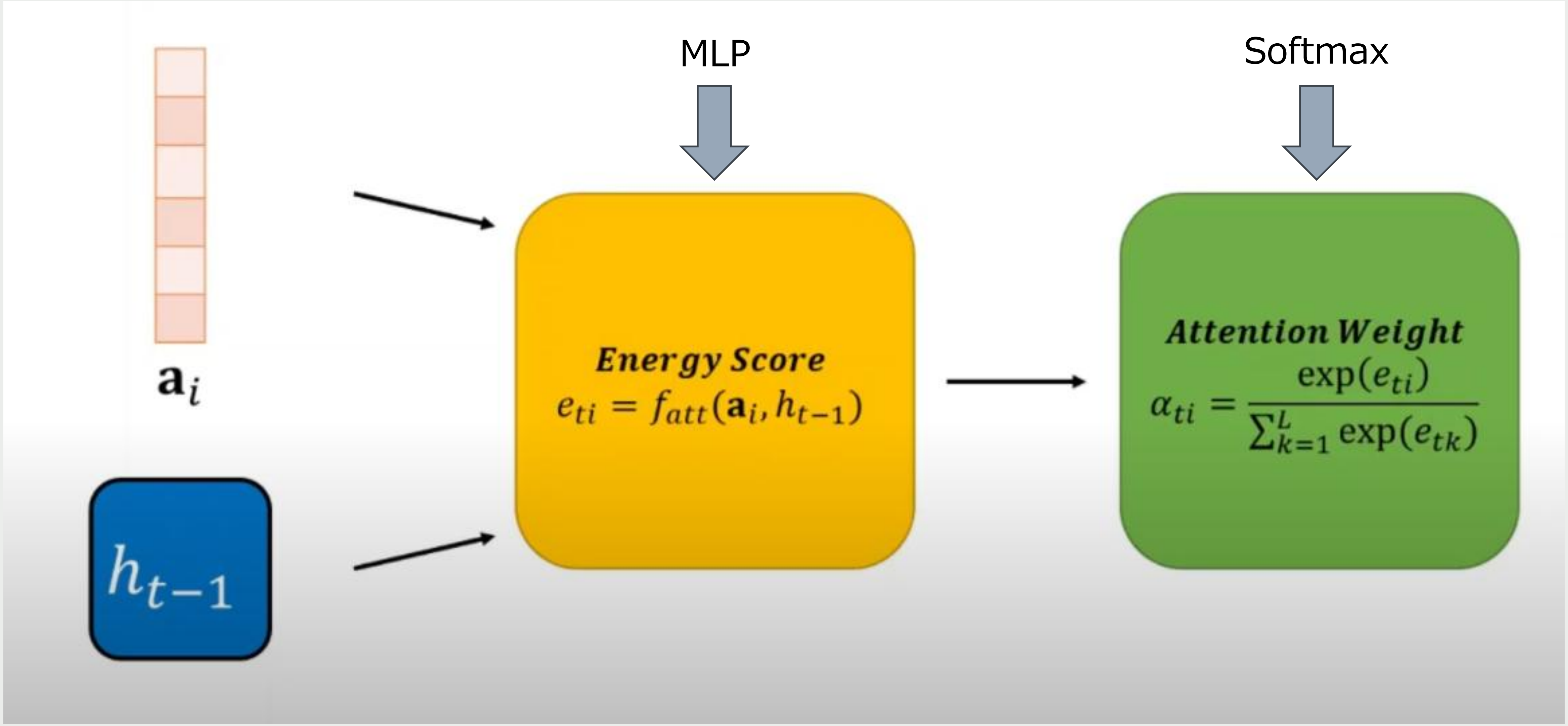
How is context vector calculated?



How is context vector calculated?



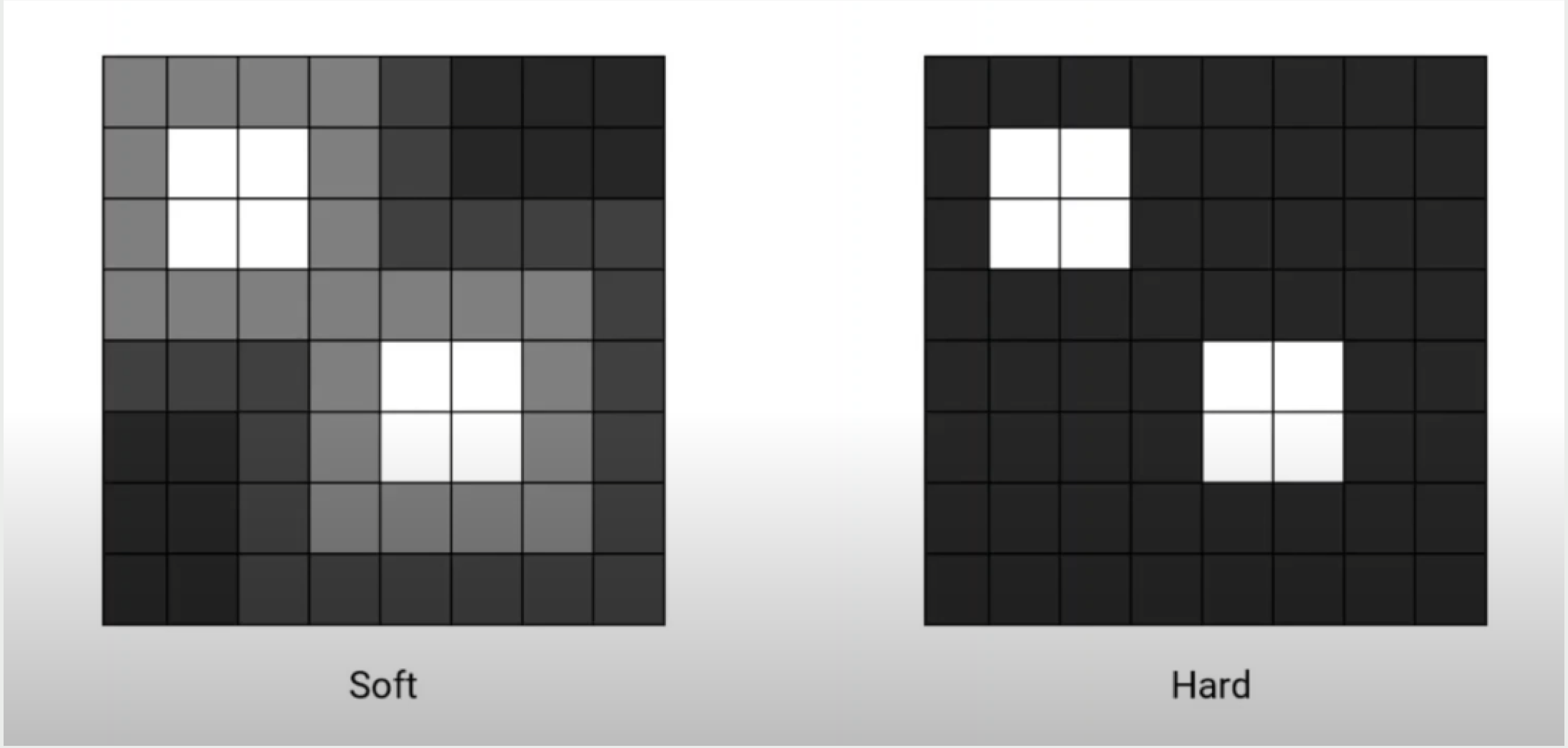
How is context vector calculated?



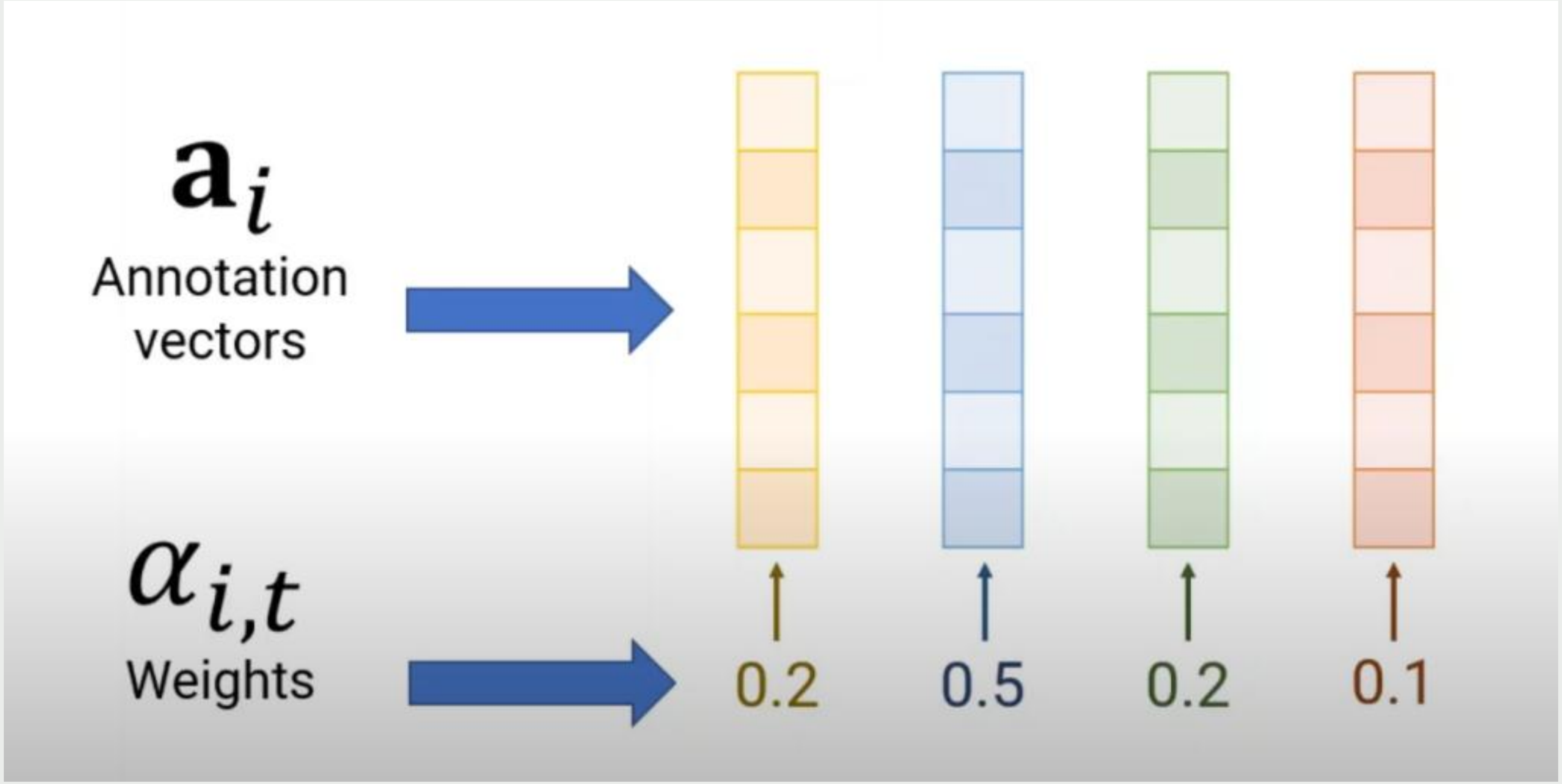
How is context vector calculated?

$$\hat{z}_t = \Phi\left(\left\{\underset{\substack{\uparrow\\ \text{Annotation}\\ \text{vector}}}{\mathbf{a}_i}\right\}, \left\{\underset{\substack{\uparrow\\ \text{Attention}\\ \text{weight}}}{\alpha_i}\right\}\right)$$

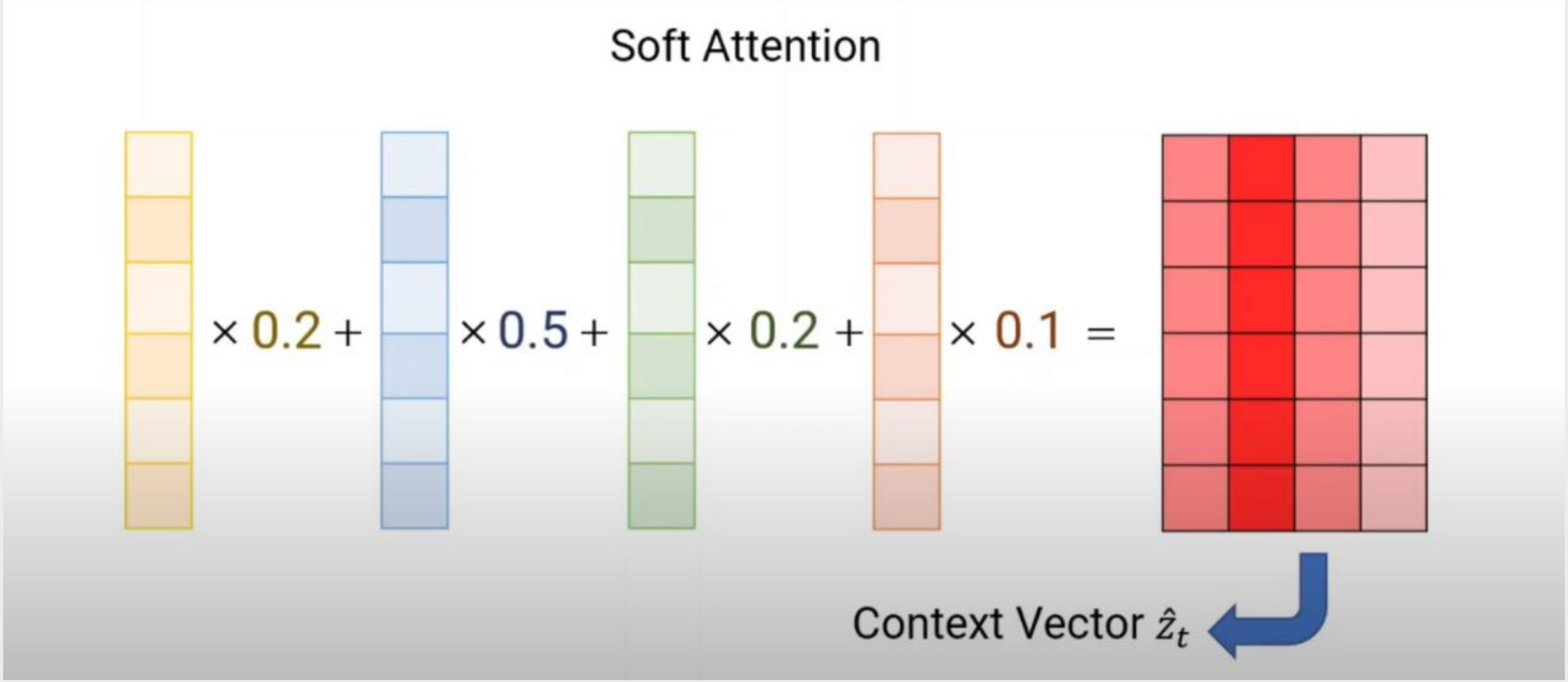
Hard and soft attention



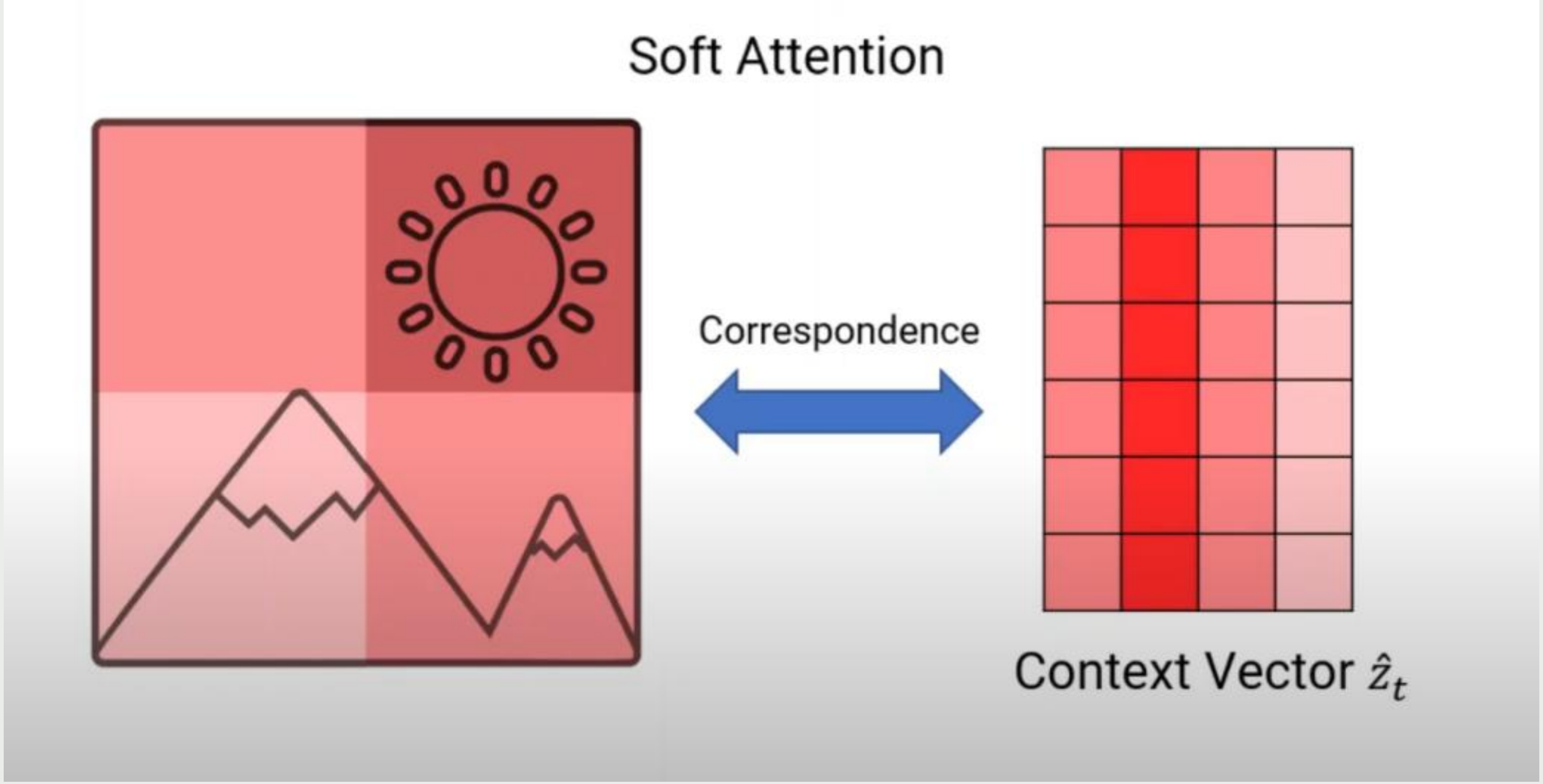
Hard and soft attention



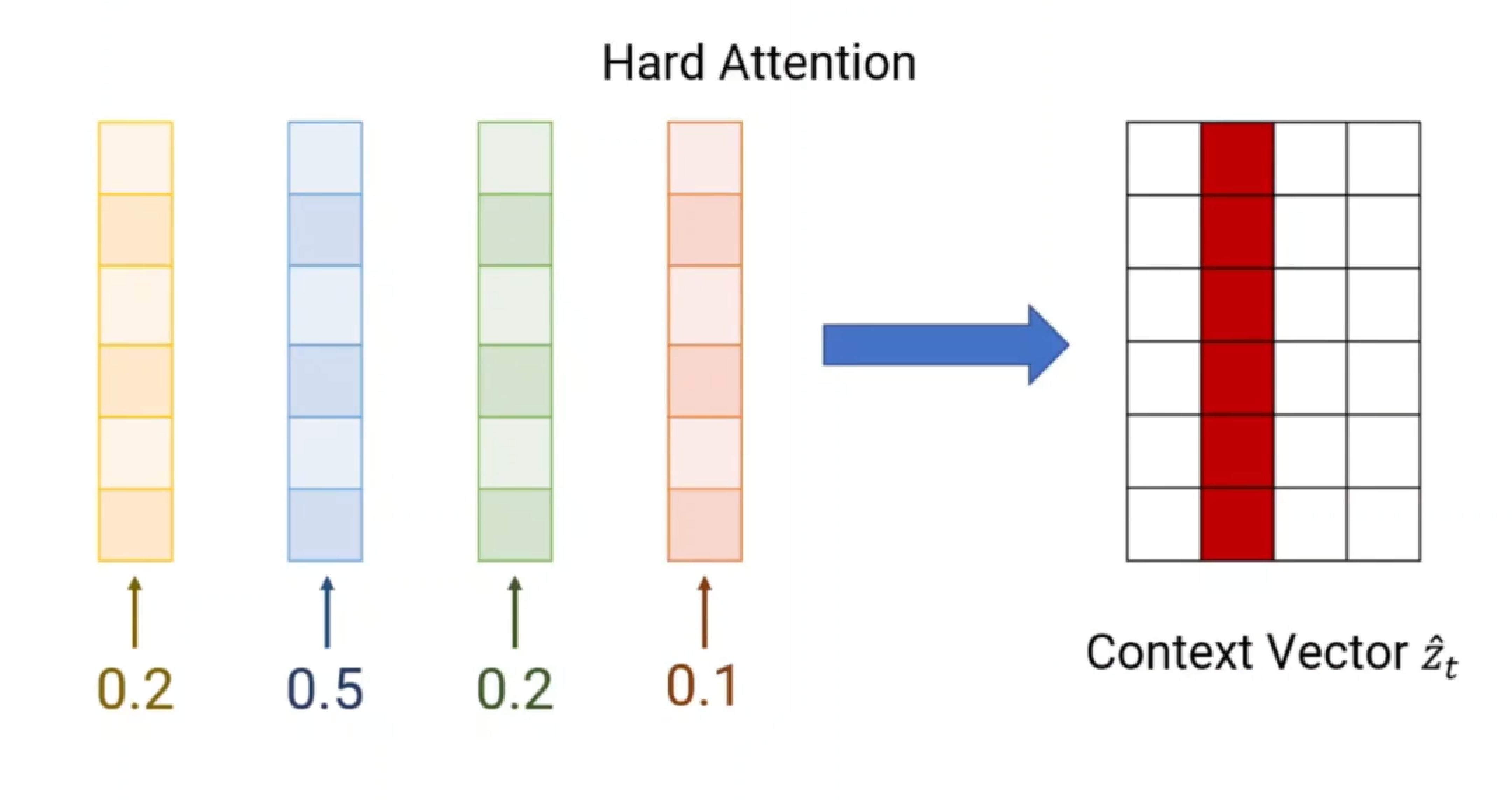
Hard and soft attention



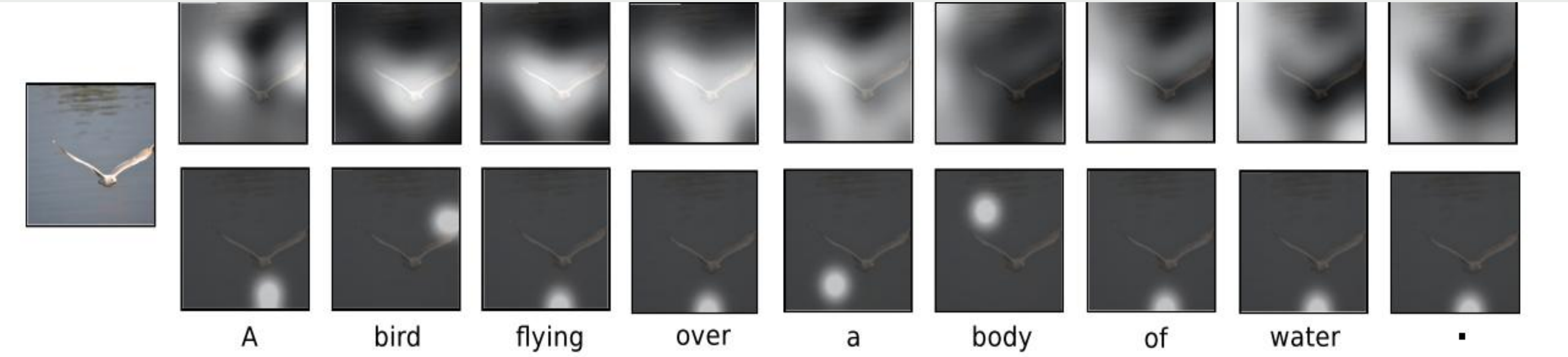
Hard and soft attention



Hard and soft attention



Hard and soft attention



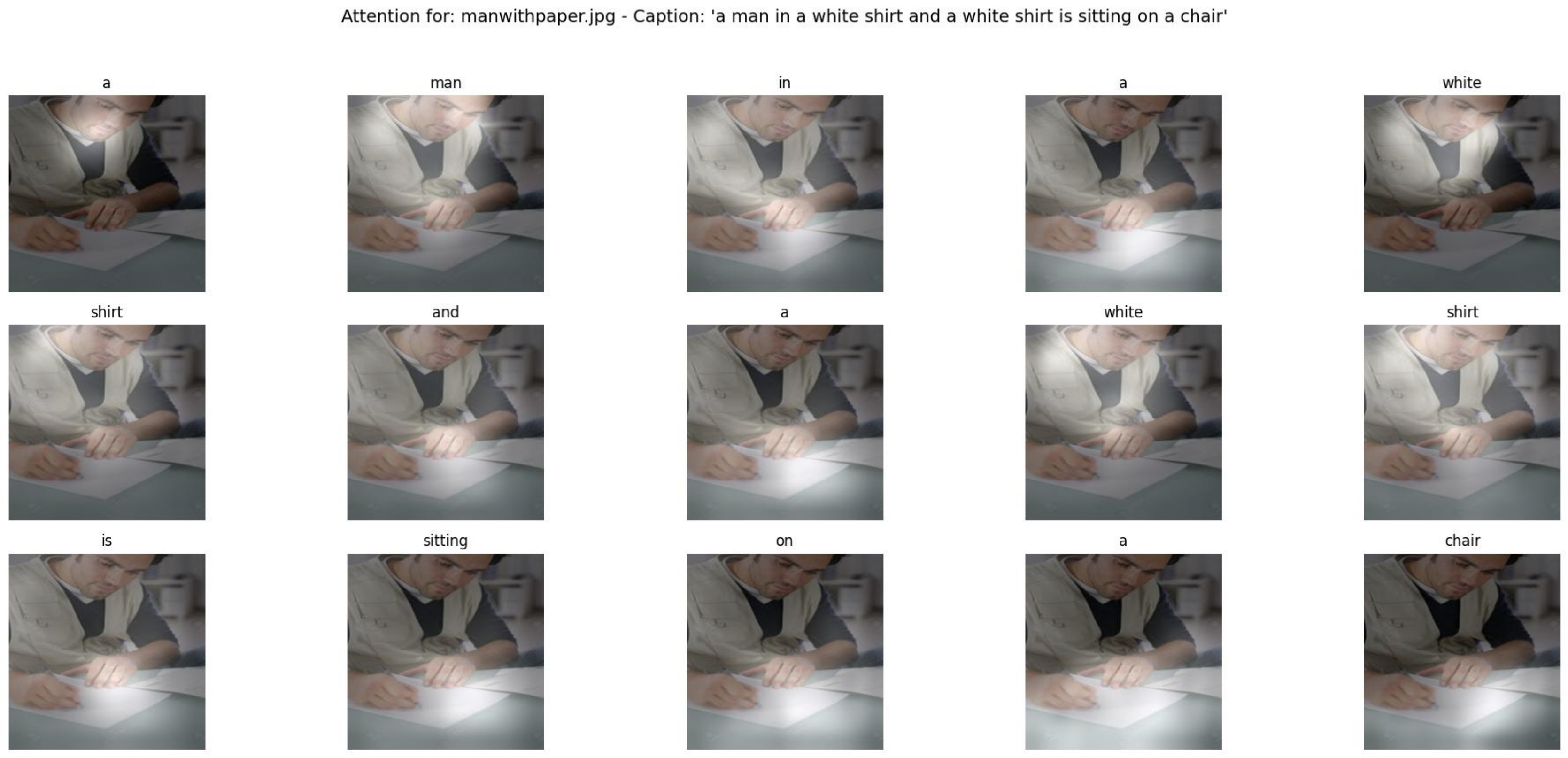
Soft

Hard

How Beam Search Works

Goal: Convert an image into a coherent and descriptive text caption.
The model:
CNN: Extracts visual features from the image.
LSTM: Generates words one by one, using these features and its internal memory.

The Problem: At each step, the LSTM outputs probabilities for *thousands* of possible next words. How do we choose the *best* sequence of words to form the entire caption?



How Beam Search Works

Easiest approach: At each step, simply pick the word with the *highest probability*.
Step 1: Predicts "A" (prob=0.9)

Step 2: Predicts "dog" (prob=0.8)

Step 3: Predicts "is" (prob=0.7)

...

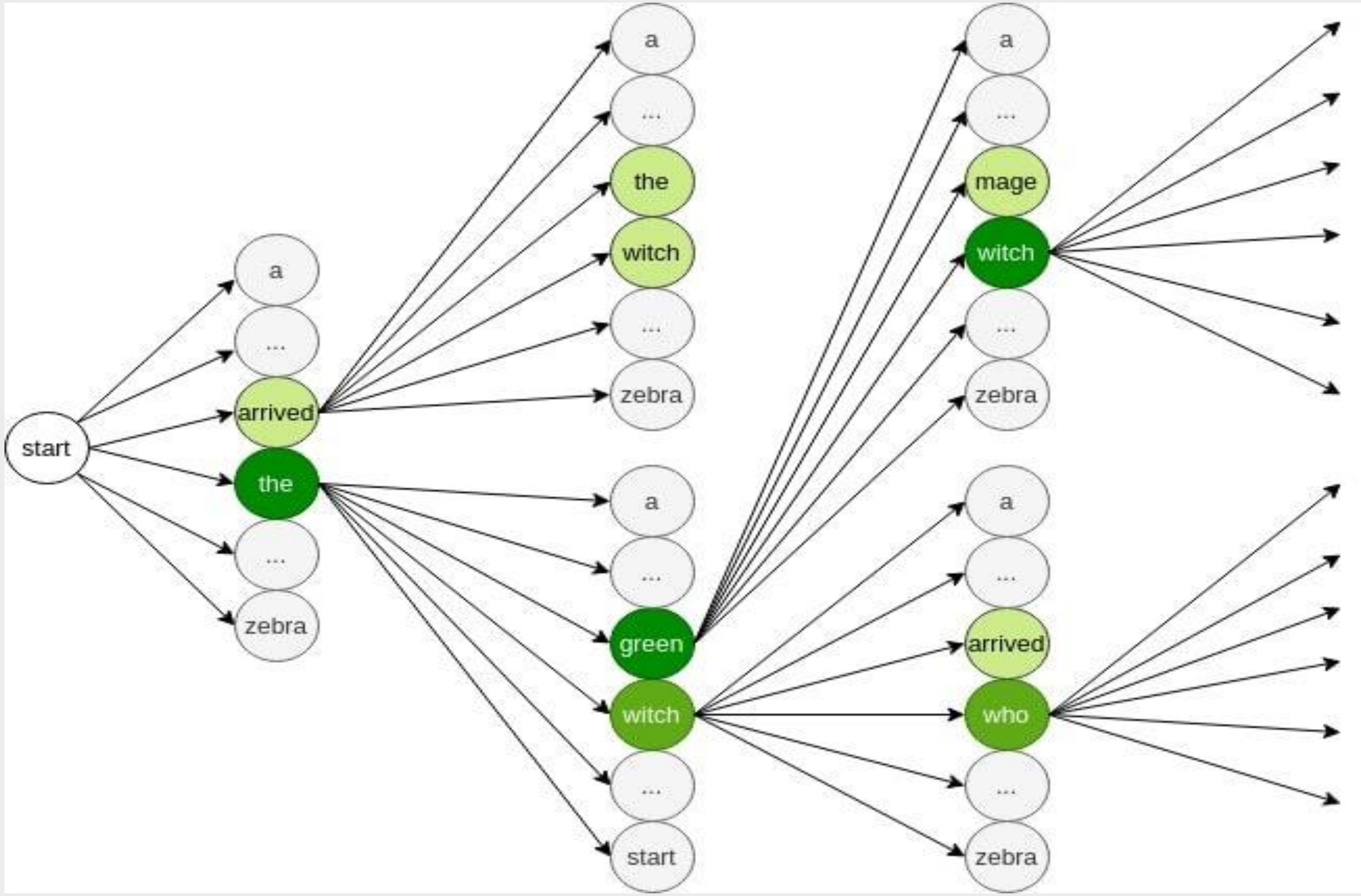
Caption: "A dog is running in the park."

Why it can fail: A locally optimal choice (the highest probability word at one step) doesn't always lead to a globally optimal (best overall) sentence.

Example: "A ***dog*** *is running" might seem best now, but "A* ***very cute*** *dog is running" might have been achievable if we hadn't been so "greedy" early on.*

How Beam Search Works

Beam search approach: Instead of just picking the single best word at each step, Beam Search explores *multiple* promising partial sequences (hypotheses) simultaneously



How Beam Search Works

Beam search approach: Instead of just picking the single best word at each step, Beam Search explores *multiple* promising partial sequences (hypotheses) simultaneously

The "Beam Size" Parameter (k)

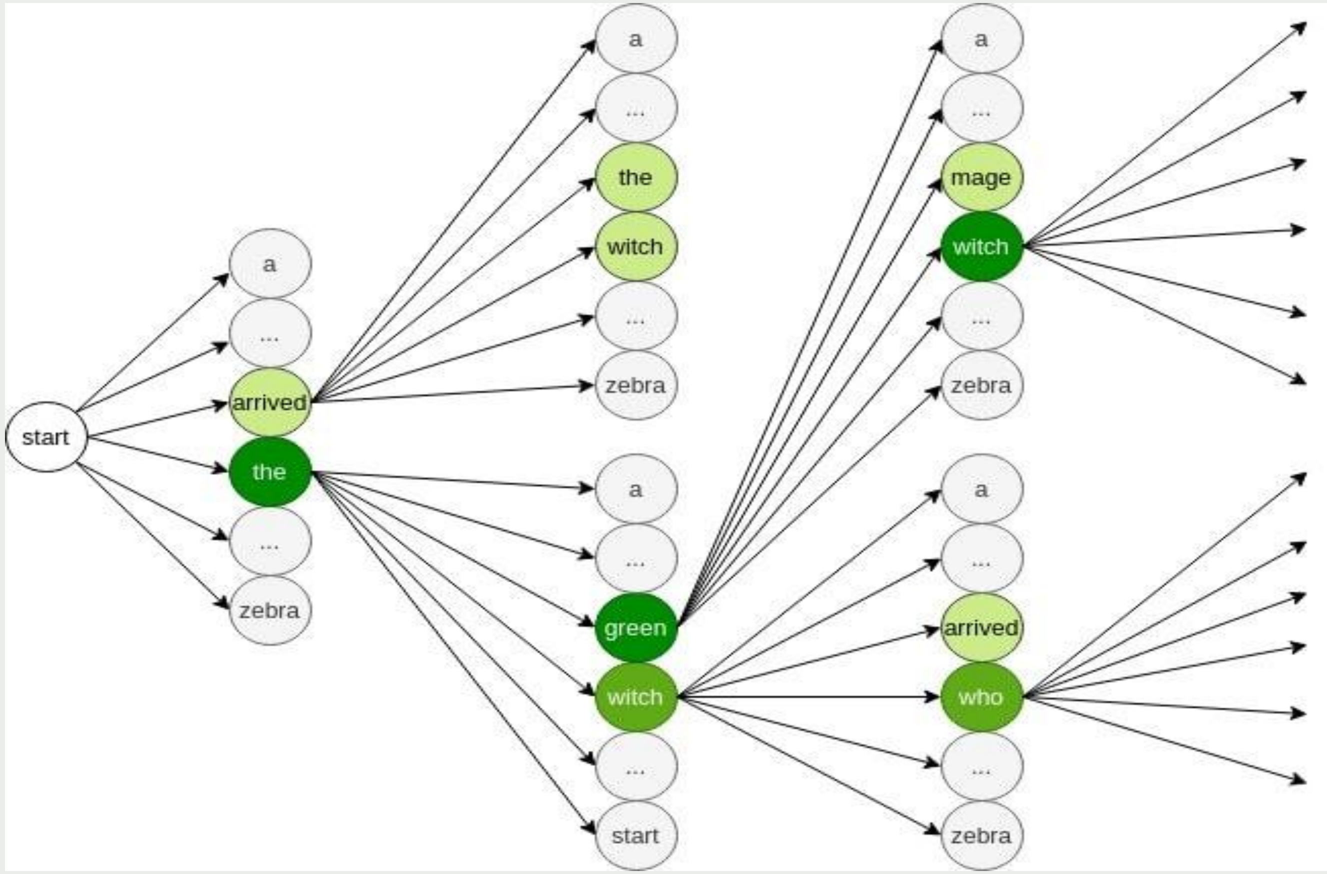
Definition: k is the maximum number of partial sequences (hypotheses) that Beam Search keeps track of at each step.

Impact of k:

k=1: This is equivalent to Greedy Search.

Small k (2-5): Explores a few options.

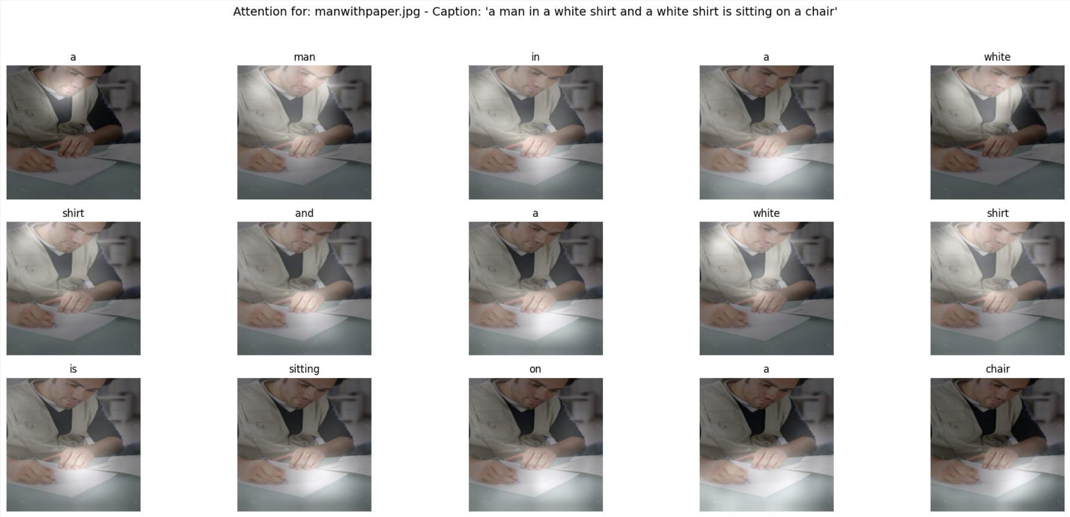
Large k (5-35): Explores a wider range of options.



How Beam Search Works

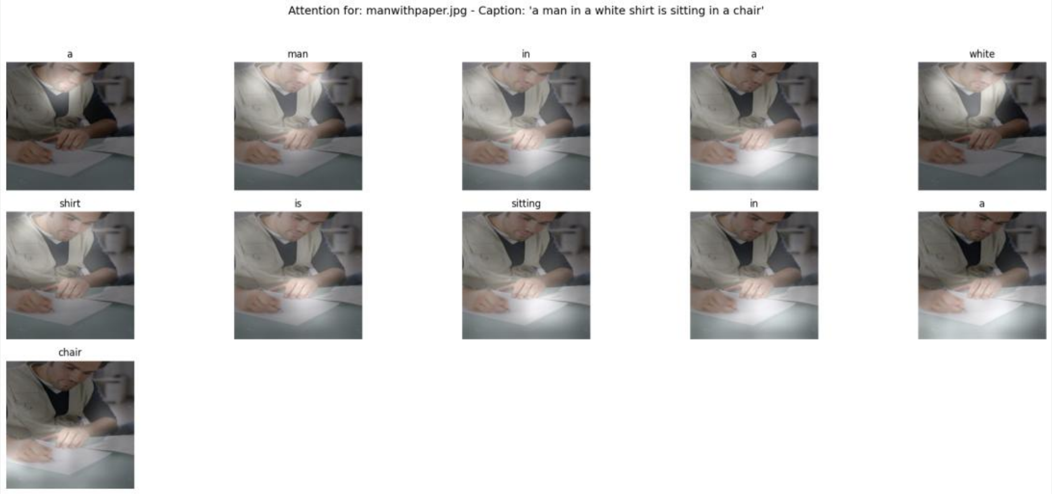
Beam search approach: Instead of just picking the single best word at each step, Beam Search explores *multiple* promising partial sequences (hypotheses) simultaneously

a man in a white shirt and a white shirt is sitting on a chair



Beam Size: 5

a man in a white shirt is sitting in a chair



Beam Size: 35